Cisco DCNM Web Services API Guide, Release 4.2
August 14, 2009

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel:  408 526-4000
     800 553-NETS (6387)
Fax:  408 527-0883

Text Part Number: OL-20088-01
The specifications and information regarding the products in this manual are subject to change without notice. All statements, information, and recommendations in this manual are believed to be accurate but are presented without warranty of any kind, express or implied. Users must take full responsibility for their application of any products.

The software license and limited warranty for the accompanying product are set forth in the information packet that shipped with the product and are incorporated herein by this reference. If you are unable to locate the software license or limited warranty, contact your Cisco representative for a copy.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB’s public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

Notwithstanding any other warranty herein, all document files and software of these suppliers are provided “as is” with all faults. Cisco and the above-named suppliers disclaim all warranties, expressed or implied, including, without limitation, those of merchantability, fitness for a particular purpose and noninfringement or arising from a course of dealing, usage, or trade practice.

In no event shall Cisco or its suppliers be liable for any indirect, special, consequential, or incidental damages, including, without limitation, lost profits or loss or damage to data arising out of the use or inability to use this manual, even if Cisco or its suppliers have been advised of the possibility of such damages.

CCDE, CCENT, CCSI, Cisco Eos, Cisco HealthPresence, Cisco IronPort, the Cisco logo, Cisco Lumin, Cisco Nexus, Cisco Nurse Connect, Cisco StackPower, Cisco StadiumVision, Cisco TelePresence, Cisco Unified Computing System, Cisco WebEx, DCE, Flip Channels, Flip for Good, Flip Mino, Flip Video, Flip Video (Design), Flipshare (Design), Flip Ultra, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn, Cisco Store, and Flip Gift Card are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, IronPort, the IronPort logo, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0907R)

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

Cisco DCNM Web Services API Guide, Release 4.2
© 2008 Cisco Systems, Inc. All rights reserved.
CONTENTS

Preface xxxiii
  Audience xxxiii
  Organization xxxiii
  Document Conventions xxxiv
  Related Documentation xxxv
  Obtaining Documentation and Submitting a Service Request i-xxxv

CHAPTER 1
Overview 1-1
  Information About Web Services 1-1
    SOAP Messages 1-1
      SOAP Message to Create Session ID 1-2
      SOAP Message to Make API Call 1-2
      Error Handling 1-6
    Licensing Requirements for Web Services 1-6
    Additional References 1-6
    Related Documents 1-6
    Standards 1-6

CHAPTER 2
AaaApp Service 2-1
  Information About AaaApp Service 2-1
  bindNetworkInterfaceToAaaServerGroups 2-1
  bindNetworkInterfaceToGlobalRadiusServerSetting 2-2
  bindNetworkInterfaceToGlobalTACACSServerSetting 2-3
  createAaaServerGroups 2-3
  createAccountingRulesInNetworkElement 2-4
  createAuthenticationRulesInNetworkElement 2-5
  createAuthorizationRulesInNetworkElement 2-5
  createGlobalAaaServers 2-6
  deleteAaaServerGroups 2-7
  deleteAccountingRules 2-7
  deleteAuthenticationRules 2-8
  deleteAuthorizationRules 2-8
Contents

Send document comments to nexus7k-docfeedback@cisco.com

deleteGlobalAaaServers 2-9
disableAaa 2-9
disableTACACS 2-10
enableAaa 2-10
enableTACACS 2-11
getAaaServerGroups 2-11
getAaaServerGroupsInNetworkElement 2-12
getAaaStateOfNetworkElements 2-12
getAccountingRules 2-13
getAccountingRulesInNetworkElement 2-13
getAuthenticationRules 2-14
getAuthenticationRulesInNetworkElement 2-14
getAuthorizationRules 2-15
getAuthorizationRulesInNetworkElement 2-15
getGlobalAaaServerSettings 2-16
getGlobalAaaServers 2-16
getGlobalAaaServersForGroupAaaServers 2-16
getGlobalAaaServersInNetworkElement 2-17
getGlobalRadiusServerSettings 2-17
getGlobalRadiusServersInNetworkElement 2-18
getGlobalTACACSServerSettings 2-18
getGlobalTACACSServersInNetworkElement 2-19
getRadiusServerGroupsInNetworkElement 2-19
getTACACSServerGroupsInNetworkElement 2-20
getAddressOfNetworkElements 2-20
modifyAaaServerGroups 2-21
modifyAccountingRules 2-21
modifyAuthenticationRules 2-22
modifyAuthorizationRules 2-23
modifyGlobalAaaServerSetting 2-23
modifyGlobalAaaServers 2-24
unbindNetworkInterfaceFromAaaServerGroups 2-24
unbindNetworkInterfaceFromGlobalRadiusServerSettings 2-25
unbindNetworkInterfaceFromGlobalTACACSServerSettings 2-25
CHAPTER 3

AclApp Service 3-1

Information About AclApp Service 3-1
addRedirectNetworkInterfacesToVlanAccessMapEntry 3-2
bindIpV4AclToNetworkInterfaces 3-2
bindIpV4AclsToVlanAccessMapEntry 3-3
bindIpV6AclToNetworkInterfaces 3-3
bindIpV6AclsToVlanAccessMapEntry 3-4
bindMacAclToNetworkInterfaces 3-4
bindMacAclsToVlanAccessMapEntry 3-5
bindTimeRangeToAces 3-5
bindVlanAccessMapToVlans 3-6
createExtendedIpAcls 3-6
createIpV6Acls 3-7
createMacAcls 3-8
createRbaclPolicies 3-9
createRbacl 3-9
createStandardIpAcls 3-10
createTimeRanges 3-11
createVlanAccessMaps 3-12
deleteExtendedIpAcls 3-12
deleteIpV6Acls 3-13
deleteMacAcls 3-13
deleteRbaclPolicies 3-14
deleteRbacl 3-14
deleteStandardIpAcls 3-15
deleteTimeRanges 3-15
deleteVlanAccessMaps 3-16
getAssociatedVlanAccessMap 3-16
getExtendedIpAclToNetworkInterfaceAssociationsInNetworkElement 3-17
getExtendedIpAcls 3-17
getExtendedIpAclsInNetworkElement 3-18
getExtendedNamedIpAclsInNetworkElement 3-18
getExtendedNumberedIpAclsInNetworkElement 3-19
getIpAclToNetworkInterfaceAssociationsInNetworkElement 3-19
getIpAclsInNetworkElement 3-19
Send document comments to nexus7k-docfeedback@cisco.com

goIPv4AclToNetworkInterfaceAssociationsInNetworkElement 3-20
goIPv4Acls 3-20
goIPv4AclsInNetworkElement 3-21
goIPv4AclsWithoutAcesInNetworkElement 3-22
goIPv6AclToNetworkInterfaceAssociationsInNetworkElement 3-22
goIPv6Acls 3-22
goIPv6AclsInNetworkElement 3-23
goIPv6AclsWithoutAcesInNetworkElement 3-23
goMacAclToNetworkInterfaceAssociationsInNetworkElement 3-24
goMacAcls 3-24
goMacAclsInNetworkElement 3-25
goMacAclsWithoutAcesInNetworkElement 3-25
goNamedIpv4AclsInNetworkElement 3-25
goNumberedIpv4AclsInNetworkElement 3-26
goRbacPolicies 3-26
goRbacPoliciesInNetworkElement 3-27
goRbacls 3-27
goRbaclsInNetworkElement 3-28
goStandardIpv4AclToNetworkInterfaceAssociationsInNetworkElement 3-28
goStandardIpv4Acls 3-29
goStandardIpv4AclsInNetworkElement 3-29
goStandardNamedIpv4AclsInNetworkElement 3-29
goStandardNumberedIpv4AclsInNetworkElement 3-30
goTimeRanges 3-30
goTimeRangesInNetworkElement 3-31
goTimeRangesWithoutEntriesInNetworkElement 3-31
goVlanAccessLogSettingInNetworkElements 3-31
goVlanAccessMaps 3-32
goVlanAccessMapsInNetworkElement 3-32
goVlanAccessMapsWithoutVlanAccessMapEntriesInNetworkElement 3-33
modifyAclSequence 3-33
modifyExtendedIpv4Acls 3-34
modifyIpv6Acls 3-35
modifyMacAcls 3-35
modifyRbacPolicies 3-36
modifyRbac 3-37
modifyStandardIpAcls 3-37
modifyTimeRanges 3-38
modifyVlanAccessLogSetting 3-39
modifyVlanAccessMaps 3-39
unbindIpv4AclFromNetworkInterface 3-40
unbindIpv4AclsFromVlanAccessMapEntry 3-40
unbindIpv6AclFromNetworkInterface 3-41
unbindIpv6AclsFromVlanAccessMapEntry 3-41
unbindMacAclFromNetworkInterface 3-41
unbindMacAclsFromVlanAccessMapEntry 3-42
unbindNetworkInterfacesFromVlanAccessMapEntry 3-42
unbindTimeRangeFromAces 3-43
unbindVlanAccessMapFromVlans 3-43

CHAPTER 4
CollectorApp Service 4-1
Information About CollectorApp Service 4-1
addStatisticalEntry 4-1
createCollector 4-2
deleteCollectionInfo 4-3
deleteStatisticalData 4-3
deleteStatisticalDataTillDate 4-3
getAllCollectionInfo 4-4
gGetAllCollectionInfoByConfigIds 4-4
gGetAllCollectionInfoByStatus 4-4
gGetAllCollectionInfoByUserld 4-5
gGetAllCollectionInfoInDevice 4-5
gGetAllCollectionInfoInDeviceByStatus 4-5
gGetAllStatisticalEntriesForDevice 4-6
gGetCollectedData 4-6
gGetCollectedStatisticalData 4-6
gGetCollectionInfo 4-7
gGetLatestCollectedStatisticalData 4-7
gGetValidConfigInstanceId 4-7
modifyCollectorAction 4-8
modifySchedule 4-8
removeStatisticalEntry 4-9
startCollector 4-9
startCollectorBySchedule 4-10
stopCollector 4-10

CHAPTER 5
CollectorExtendedApp Service 5-1
Information About CollectorExtendedApp Service 5-1
getCollectedStatisticalData 5-1
getCollectedStatisticalDataByDaysMaxLimit 5-1
getCollectedStatisticalDataByMaxLimit 5-2
getCollectedStatisticalDataByTimeRange 5-2
getCollectedStatisticalDataByTimeRangeInterval 5-3
getCollectedStatisticalDataByTimeRangeMaxLimit 5-3
getCollectedStatisticalDataSizeByTimeRangeInterval 5-4
getPreviousCollectedData 5-4

CHAPTER 6
ConfigArchiveApp Service 6-1
Information About ConfigArchiveApp Service 6-1
createConfigArchiveTask 6-1
deleteConfigArchiveStatusHistoryTillDateForTask 6-2
deleteConfigArchiveTasks 6-2
deleteConfigFilesTillDate 6-2
deleteRollbackStatusHistoryTillDateForDevice 6-2
fetchCfgDiff 6-3
fetchNetworkElementConfigData 6-3
getAllConfigTasks 6-4
gerArchivalDeviceStatusEntries 6-4
gerArchivedNetworkElementConfigData 6-4
gerConfigArchiveSettings 6-4
gerConfigArchiveStatusHistoryForTask 6-5
gerConfigVersions 6-5
gerRollbackStatusHistoryForDevice 6-5
modifyConfigArchiveSettings 6-6
modifyConfigArchiveTask 6-6
modifyConfigVersion 6-6
DaiApp Service  7-1

Information About DaiApp Service  7-1
bindArpAclOnVlans  7-1
bindArpAclOnVlansByArpAclName  7-2
bindArpAclOnVlansForRange  7-2
clearArpRateLimitingConfigurationInInterfaces  7-3
createArpAcls  7-3
deleteArpAcls  7-4
disableDaiOnVlans  7-4
enableDaiOnVlans  7-5
enableDaiOnVlansByRange  7-5
getArpAclsWithoutAcesInNetworkElement  7-6
generateArpAcls  7-6
generateArpAclsInVlans  7-6
generateArpAclsWithoutAcesInNetworkElement  7-6
generateDaiDisabledVlansInNetworkElement  7-7
generateDaiEnabledVlansInNetworkElement  7-7
generateDaiGlobalSettingsInNetworkElements  7-7
generateDaiSettingOnVlans  7-8
generateInterfacesWithArpRateLimitingInNetworkElement  7-8
genrateTrustStateSettingInInterfaces  7-8
genrateUntrustedInterfacesWithDefaultRateInNetworkElement  7-9
genrateVlansWithDaiSettingNetworkElement  7-9
modifyAclSequence  7-9
modifyArpAcls  7-10
modifyDaiGlobalSettingsInNetworkElements  7-10
modifyDaiOnVlans  7-11
modifyDaiSettingsAndArpAclBindingsOnVlans  7-11
modifyTrustStateSettings  7-12
unbindArpAclFromVlans  7-12
**CHAPTER 8**

**DataPurgingApp Service**

- Information About DataPurgingApp Service 8-1
- createDataPurgingSchedule 8-1
- deleteDataPurgingSchedule 8-2
- getDataPurgingSchedule 8-2
- modifyDataPurgingSchedules 8-2
- purgeDataOnDemand 8-3

**CHAPTER 9**

**DeviceListApp Service**

- Information About DeviceListApp Service 9-1
- addNetworkElements 9-1
- bindNetworkElementsToGroup 9-1
- createNetworkElementGroup 9-2
- deleteAllNetworkElementGroups 9-2
- deleteAllNetworkElements 9-2
- deleteNetworkElementGroups 9-2
- deleteNetworkElements 9-3
- discover 9-3
- discoverNetworkElements 9-3
- getAllDiscoveryTasks 9-4
- getAllNetworkElementGroups 9-4
- getAllNetworkElements 9-4
- getDiscoveryStatus 9-4
- getNetworkElementGroups 9-5
- incrementalDiscovery 9-5
- modifyNetworkElementGroups 9-5
- performDiscovery 9-5
- performIncrementalDiscovery 9-6
- performNetworkElementsDiscovery 9-6
- performNetworkElementsRediscovery 9-6
- performNetworkElementsUnmanaged 9-7
- rediscoverNetworkElements 9-7
- unbindNetworkElementsToGroup 9-7
- unmanageNetworkElements 9-7
CHAPTER 10 DhcpSnoopingApp Service

Information About DhcpSnoopingApp Service 10-1
addStaticBindings 10-1
clearRateLimitingConfigurationInInterfaces 10-2
deleteAllBindings 10-2
deleteAllBindingsInNetworkElements 10-3
deleteAllDynamicBindingsInInterfaces 10-3
deleteAllDynamicBindingsInNetworkElements 10-3
deleteAllDynamicBindingsInVlanOfAnInterface 10-4
deleteAllStaticBindingsInInterfaces 10-4
deleteAllStaticBindingsInNetworkElements 10-5
deleteAllStaticBindingsInVlanOfAnInterface 10-5
deleteDynamicBindings 10-6
deleteStaticAndDynamicBindings 10-6
deleteStaticBindings 10-7
disableDhcpSnoopingInNetworkElement 10-7
disableDhcpSnoopingOnVlans 10-7
disableDhcpSnoopingService 10-8
enableDhcpSnoopingOnVlans 10-8
enableDhcpSnoopingOnVlansByRange 10-8
enableDhcpSnoopingService 10-9
getcDhcpSnoopingDisabledVlansInNetworkElement 10-9
getcDhcpSnoopingEnabledVlansInNetworkElement 10-10
getcDhcpSnoopingGlobalSettingOnNetworkElements 10-10
getcDhcpSnoopingServiceStateInNetworkElements 10-10
getcDhcpSnoopingSettingOnVlans 10-11
getcDynamicBindingsInInterface 10-11
getcInterfacesWithDhcpRateLimitingInNetworkElement 10-12
getcInterfacesWithDynamicBindingsInNetworkElement 10-12
getcInterfacesWithStaticBindingsInNetworkElement 10-13
getcNumberOfDynamicBindingsInVlans 10-13
getcNumberOfStaticBindingsInVlans 10-13
getcStaticBindingsInInterface 10-14
getcStaticBindingsInVlanOfAnInterface 10-14
getcStaticDhcpSnoopingBindings 10-15
getTrustStateSettingOnInterfaces 10-15
getUntrustedInterfacesWithDefaultRateLimitInNetworkElement 10-16
getVlansWithDhcpSnoopingInNetworkElement 10-16
modifyDhcpSnoopingGlobalSetting 10-16
modifyStaticBindings 10-17
modifyTrustStateSettings 10-17

CHAPTER 11
Dot1xApp Service 11-1
Information About Dot1xApp Service 11-1
disableDot1x 11-1
disableDot1xInInterfaces 11-2
disableDot1xService 11-2
enableDot1x 11-2
enableDot1xInInterfaces 11-3
enableDot1xService 11-3
getDot1xConflictingPorts 11-4
getDot1xGlobalSetting 11-4
getDot1xNetworkInterfacesSetting 11-4
getDot1xNetworkInterfacesInNetworkElement 11-5
modifyDot1xGlobalSetting 11-6
modifyDot1xNetworkInterfacesSetting 11-6

CHAPTER 12
EtherChannelApp Service 12-1
Information About EtherChannelApp Service 12-1
addPortsToEtherChannelEndPoint 12-1
createEtherChannel 12-2
createEtherChannelForSpecifiedEndpoints 12-3
deleteEtherChannelEndPoints 12-4
deleteEtherChannels 12-4
disableLacp 12-5
enableLacp 12-5
getAllEtherChannels 12-6
getEtherChannelGlobalSettings 12-6
getEtherChannelLinks 12-7
getEtherChannelsInNetworkElement 12-7
Send document comments to nexus7k-docfeedback@cisco.com

getLacpStateOfNetworkElements 12-8
getNetworkCardsWithLoadBalanceSettings 12-8
modifyEtherChannelGlobalSetting 12-9
modifyEtherChannelInterfaces 12-9
modifyNetworkCardEtherChannelLoadBalanceSettings 12-10
modifyNetworkInterfaceLinks 12-10
removePortsFromEtherChannelEndPoint 12-11

CHAPTER 13
EventApp Service 13-1
Information About EventApp Service 13-1
createUserAction 13-1
deleteEventsBeforeATimeStamp 13-1
deleteEventsBetweenTimestamps 13-1
getEventCountAfterATimestamp 13-2
getEventsAfterATimestamp 13-2
getEventsBeforeATimestamp 13-2
getEventsBetweenTimestamps 13-3
getEventsBetweenTimestampsByEventClass 13-3
getEventsForEventType 13-4
getEventsForEventTypeByEventClass 13-4
getEventsForFeature 13-4
getEventsForFeatureByEventClass 13-5
getEventsForPerceivedSeverity 13-5
getEventsForPerceivedSeverityByEventClass 13-5
ggetLastEvent 13-6
getUserActions 13-6
modifyEventStatus 13-6

CHAPTER 14
GlbpApp Service 14-1
Information About GlbpApp Service 14-1
addGlbpGroupSetting 14-1
bindTrackingObjectsToGlbpGroupSetting 14-2
createGlbpGroupSettings 14-2
disableService 14-3
enableService 14-3
getActiveTimers 14-4
getAllGlbpGroupSettings 14-4
getAllGlbpGroupsInNetwork 14-4
getForwarderStatus 14-5
getGlbpAuthenticationSetting 14-5
getGlbpGatewayStatus 14-6
getGlbpGlobalSetting 14-6
getGlbpGroupIdsInNetworkInterface 14-6
getGlbpGroupSetting 14-6
getGlbpGroupSettingsInNetworkElement 14-7
getGlbpGroupSettingsInNetworkInterface 14-7
getGlbpTimers 14-8
getGlbpVirtualForwarderSetting 14-8
getGlbpWeightingSetting 14-8
getTrackedObjects 14-9
modifyAuthenticationSetting 14-9
modifyGlbpGroupSetting 14-10
modifyTimers 14-10
modifyWeightingSetting 14-11
removeGlbpGroupSetting 14-11
removeGlbpGroupSettingByGroupld 14-11
unbindTrackingObjectsFromGlbpGroupSetting 14-12

CHAPTER 15
HsrpApp Service 15-1
Information About HsrpApp Service 15-1
addHsrpGroupSetting 15-1
bindTrackingObjectToHsrpGroupSetting 15-2
disableHsrpService 15-2
enableHsrpService 15-3
getHsrpAuthenticationSetting 15-3
getHsrpGlobalSetting 15-4
getHsrpGroupSetting 15-4
getHsrpGroupSettingsInNetworkElement 15-4
getHsrpRouterStatus 15-5
getHsrpTimers 15-5
getIpInterfaceWithHsrpSettings 15-5
getTrackingObject 15-6
modifyAuthenticationSetting 15-6
modifyHsrpGroupSetting 15-6
modifyHsrpTimers 15-7
modifyIpInterfaceWithHsrpSettings 15-7
removeHsrpGroupSetting 15-8
unbindTrackingObjectFromHsrpGroupSetting 15-8

CHAPTER 16

InterfacesApp Service 16-1
Information About InterfacesApp Service 16-1
createLoopbackNetworkInterfaces 16-1
createRoutedSubNetworkInterfaces 16-2
createVlanNetworkInterfaces 16-3
deleteLogicalNetworkInterfaces 16-3
disableSviService 16-4
disableUdldService 16-4
enableSviService 16-4
enableUdldService 16-5
fetchCurrentTransceiverStatusForPorts 16-5
getAclAssociations 16-6
getActiveRoutedPortsCountInNetworkElement 16-6
getActiveSwitchedPortsCountInNetworkElement 16-6
getAdminDownRoutedPortsCountInNetworkElement 16-7
getAdminDownSwitchedPortsCountInNetworkElement 16-7
getAllInterfacesInNetworkElement 16-7
getAddressDetectionSettings 16-8
getAddressRecoverySettings 16-9
getIpAclAssociationsForNetworkInterface 16-9
getLogicalInterfacesInNetworkElement 16-9
getLoopbackInterfacesInNetworkElement 16-10
getMacAclAssociationsForNetworkInterface 16-10
getMissingPortGroupMembers 16-11
getMgmtInterfaces 16-11
getNetworkInterfaceSettings 16-12
getNetworkInterfaces 16-12
getNetworkInterfacesByName 16-13
getAddressDownRoutedPortsCountInNetworkElement 16-13
getOperDownSwitchedPortsCountInNetworkElement 16-13
getPortStatusSummary 16-14
getPortUsedForDiscovery 16-14
getPortsInModule 16-14
getPortsInNetworkElement 16-15
getPortsWithPortChannelAssociations 16-16
getPortsWithSpanAssociations 16-16
getPortsWithVlanAssociations 16-17
getRoutedPortsInModule 16-17
getRoutedPortsInNetworkElement 16-18
getRoutedPortsWithSecurityAssociations 16-18
getRoutedSubNetworkInterfacesInNetworkElement 16-19
getSubInterfacesForRoutedNetworkInterface 16-19
getSviServiceStateForNetworkElements 16-20
getSwitchedPortsCountInModule 16-20
getSwitchedPortsInModule 16-20
getSwitchedPortsInModuleByMode 16-21
getSwitchedPortsInNetworkElement 16-21
getSwitchedPortsWithSecurityAssociations 16-22
getTotalPortsCountInNetworkElement 16-22
getTotalRoutedPortsCountInNetworkElement 16-23
getTotalSwitchedPortsCountInNetworkElement 16-23
getTunnelInterfacesInNetworkElement 16-23
getUdldServiceStateForNetworkElements 16-24
getVlanNetworkInterfacesInNetworkElement 16-24
modifyErrorDetectionSettingInNetworkElement 16-24
modifyErrorRecoverySettingInNetworkElement 16-25
modifyNetworkInterfaceSetting 16-25
modifyNetworkInterfaces 16-26
modifyPortModeToRouted 16-27
modifyPortModeToSwitched 16-27
modifyRoutedSubNetworkInterfaces 16-28
modifySwitchPortMode 16-29

CHAPTER 17
ImageInstallApp Service 17-1

Information About ImageInstallApp Service 17-1
abortJob 17-1
abortTask 17-2
createJob 17-2
deleteJob 17-3
fetchDiskFreeSpace 17-3
fetchDiskFreeSpaceFromStandbySup 17-3
getAllInstallTasks 17-4
getAllJobs 17-4
getFailureLog 17-4
getJob 17-4
getStatusOfJobInNetworkElement 17-5
modifyJob 17-5
performVersionCompatability 17-6
resumeJob 17-6
suspendJob 17-7

CHAPTER 18
IpSourceGuardApp Service 18-1
Information About IpSourceGuardApp Service 18-1
addIpSourceBindings 18-1
deleteAllIpSourceBindingsInInterfaces 18-2
deleteAllIpSourceBindingsInNetworkElements 18-2
deleteAllIpSourceBindingsInVlanOfAnInterface 18-2
deleteIpSourceBindings 18-3
disableIpSourceGuardInNetworkElement 18-3
disableIpSourceGuardOnInterfaces 18-3
enableIpSourceGuardOnInterfaces 18-4
gerInterfacesWithIpSourceBindingsInNetworkElement 18-4
gerIpSourceBindings 18-5
gerIpSourceBindingsInInterface 18-5
gerIpSourceBindingsInVlanOfAnInterface 18-6
gerIpSourceGuardDisabledInterfacesInNetworkElement 18-6
gerIpSourceGuardEnabledInterfacesInNetworkElement 18-7
gerIpSourceGuardSettingOnInterfaces 18-7
getNumberOfDynamicBindingsInVlan 18-7
getNumberOfIpSourceBindingsInInterfaces 18-8
getNumberOfIpSourceBindingsInVlans 18-8
modifyIpSourceBindings 18-8

**CHAPTER 19**

**InventoryApp Service** 19-1

Information About InventoryApp Service 19-1
getAllChassisInNetwork 19-1
getAllNetworkElements 19-1
getAllPhysicalLinkInNetwork 19-2
getAllPhysicalLinks 19-2
getCardsInChassis 19-2
getCdpNeighbors 19-3
getChassis 19-3
fetchCurrentSensorsInCard 19-4
fetchCurrentSensorsInChassis 19-4
getFansInChassis 19-5
getFileSystemsInChassis 19-5
getNetworkCardsInChassis 19-6
getPhysicalLinks 19-6
getPhysicalPorts 19-7
getPhysicalPortsInSlot 19-7
getPowerSuppliesInChassis 19-8
getProcessorCardsInChassis 19-8
getSensorsInChassis 19-9
getServiceCardsInChassis 19-9
getSlotsInChassis 19-10
getSlotsInNetworkElement 19-10
getSoftwareIdentitiesInNetwork 19-10
getSystemUpTime 19-11

**CHAPTER 20**

**KeyChainApp Service** 20-1

Information About KeyChainApp Service 20-1
createKeyChains 20-1
deleteKeyChains 20-1
getKeyChains 20-2
getKeyChainsById 20-2
modifyKeyChains 20-3
## LicensingApp Service

21-1

Information About LicensingApp Service 21-1

getDetailedLicenseInfo 21-1

getHostIdentifier 21-1

getLicenseInfo 21-1

performActivateLicenses 21-2

## LogApp Service

22-1

Information About LogApp Service 22-1

getDefaultLogLevel 22-1

getLogLevel 22-1

log 22-1

modifyDefaultLogLevel 22-2

modifyLogLevel 22-2

## NacLpIpApp Service

23-1

Information About NacLpIpApp Service 23-1

addExceptionListHostsToIdentityProfile 23-1

bindAccessListToIdentityPolicies 23-2

bindEapOverUdpValidationToNetworkInterfaces 23-2

bindIdentityPolicyToExceptionListHosts 23-3

bindIdentityPolicyToExceptionListHostsByName 23-3

bindIpAdmissionControlRulesToNetworkInterfaces 23-4

clearIpAdmissionControlRuleFromInterfaces 23-4

createIdentityPoliciesInNetworkElement 23-5

createIdentityProfile 23-5

createIpAdmissionControlRulesInNetworkElement 23-6

deleteAllIpAdmissionControlRulesInNetworkElement 23-6

deleteAllIdentityPoliciesInNetworkElement 23-6

deleteIdentityPolicy 23-7

deleteIdentityProfileFromNetworkElement 23-7

deleteIpAdmissionControlRules 23-7

disableClientlessAuthenticationInNetworkElements 23-8

disableIpDeviceTrackingInNetworkElements 23-8

disableNacService 23-9

enableClientlessAuthenticationInNetworkElements 23-9
enableIpDeviceTrackingInNetworkElements 23-9
enableNacService 23-10
getAdmissionControlRulesInNetworkElement 23-10
getAllNacHostSessionInNetworkElement 23-10
getClientlessAuthenticationInNetworkElements 23-11
getEapOudpValidationSettingInInterfaces 23-11
getExceptionListHostsInIdentityProfile 23-12
getIdentityPoliciesInNetworkElement 23-12
getIdentityPolicyForExceptionListHost 23-12
getIdentityProfilesInNetworkElement 23-13
getInterfacesUsingIpAdmissionControlRule 23-13
getLpIpAdmissionControlRuleAppliedOnInterfaces 23-13
getLpIpDeviceTrackingInNetworkElements 23-14
getLpIpGlobalSettingsInNetworkElements 23-14
getLpIpTrackedDevicesInNetworkElement 23-14
getLpIpTrackedDevicesInSwitchedNetworkInterface 23-15
getNacHostSessionInSwitchedNetworkInterface 23-15
modifyClientlessAuthentication 23-15
modifyExceptionListHostsInIdentityProfile 23-16
modifyIdentityPolicies 23-16
modifyIdentityProfiles 23-16
modifyLpIpAdmissionControlRules 23-17
modifyLpIpDeviceTracking 23-17
modifyLpIpGlobalSettingsInNetworkElements 23-18
removeExceptionListHostsFromIdentityProfile 23-18
unbindIdentityPolicyFromExceptionListHosts 23-18

CHAPTER 24
ObjectTrackingApp Service 24-1
Information About ObjectTrackingApp Service 24-1
createTrackedObjectsInNetworkElements 24-1
deleteTrackedObjects 24-1
getTrackedClients 24-2
getTrackedObjects 24-2
getTrackedObjectsInNetworkElement 24-3
modifyTrackedObjects 24-3
CHAPTER 25  
PollerApp Service  25-1
  Information About PollerApp Service  25-1
  getAllPollerTasks  25-1
  getLoggingLevelSettings  25-1
  getLoggingLevelSettingsForDcnmDefaults  25-2
  getPollerInterval  25-2
  modifyLoggingLevelSettings  25-2
  modifyLoggingLevelSettingsToDcnmDefaults  25-2
  modifyPollerInterval  25-3
  startSyslogPoller  25-3
  stopSyslogPoller  25-3
  synchronizeWithDevice  25-3
  validateLoggingLevelSettingsForDcnmDefaults  25-4

CHAPTER 26  
PortSecurityApp Service  26-1
  Information About PortSecurityApp Service  26-1
  bindPortSecuritySettingToNetworkInterfaces  26-1
  bindStaticSecureHostsToAccessNetworkInterface  26-2
  bindStaticSecureHostsToPvlanHostNetworkInterface  26-2
  bindStaticSecureHostsToPvlanPromiscuousNetworkInterface  26-2
  bindStaticSecureHostsToVlanInTrunkNetworkInterface  26-3
  bindStaticSecureHostsToVlanInTrunkNetworkInterfaceByRange  26-3
  clearPortSecuritySettingToNetworkInterfaces  26-4
  createPortSecurityNetworkInterfaceSettingForVlans  26-4
  deletePortSecurityNetworkInterfaceSettingForVlans  26-5
  disablePortSecurityInAllInterfacesInNetworkElement  26-5
  disablePortSecurityInNetworkInterfaces  26-6
  disablePortSecurityService  26-6
  disablePortSecurityStopLearning  26-6
  enablePortSecurityInNetworkInterfaces  26-7
  enablePortSecurityService  26-7
  enablePortSecurityStopLearning  26-8
  getAllDynamicSecureMacAddressesInNetworkInterface  26-8
  getAllDynamicSecureMacAddressesInVlanOfNetworkInterface  26-8
CHAPTER 27

RbacApp Service 27-1

Information About RbacApp Service 27-1
createComponentGroups 27-1
createNetworkElementUserRoles 27-2
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>createNetworkElementUsers</td>
<td>27-2</td>
</tr>
<tr>
<td>deleteComponentGroups</td>
<td>27-3</td>
</tr>
<tr>
<td>deleteNetworkElementUserRoles</td>
<td>27-4</td>
</tr>
<tr>
<td>deleteNetworkElementUsers</td>
<td>27-4</td>
</tr>
<tr>
<td>getComponentAuths</td>
<td>27-4</td>
</tr>
<tr>
<td>getComponentAuthsInNetworkElement</td>
<td>27-5</td>
</tr>
<tr>
<td>getComponentGroups</td>
<td>27-5</td>
</tr>
<tr>
<td>getComponentGroupsInNetworkElement</td>
<td>27-6</td>
</tr>
<tr>
<td>getNetworkElementUserRoles</td>
<td>27-6</td>
</tr>
<tr>
<td>getNetworkElementUserRolesInNetworkElement</td>
<td>27-7</td>
</tr>
<tr>
<td>getNetworkElementUsers</td>
<td>27-8</td>
</tr>
<tr>
<td>getNetworkElementUsersInNetworkElement</td>
<td>27-8</td>
</tr>
<tr>
<td>modifyComponentGroups</td>
<td>27-9</td>
</tr>
<tr>
<td>modifyNetworkElementUserRoles</td>
<td>27-9</td>
</tr>
<tr>
<td>modifyNetworkElementUsers</td>
<td>27-10</td>
</tr>
</tbody>
</table>

**Chapter 28**

RepositoryApp Service  
Information About RepositoryApp Service  
abortCurrentProcess  
createConnection  
createConnectionWithRepSvr  
createRepositoryServers  
deleteConnection  
deleteConnectionUseRepSvr  
deleteRepositoryFiles  
deleteRepositoryServers  
getAllRepositoryServers  
getFirstLevelFileNames  
getRecursivelyFileNames  
getResultsFromConnection  
modifyRepositoryServers  
validateConnectivity  

**Chapter 29**

SecurityApp Service  
Information About SecurityApp Service  
getSessionId  

CHAPTER 30  ServerAdminApp Service  30-1
  Information About ServerAdminApp Service  30-1
  getServerSetting  30-1
  getServerStatus  30-1
  modifyServerSetting  30-1

CHAPTER 31  SoftwareApp Service  31-1
  Information About SoftwareApp Service  31-1
  getProduct  31-1

CHAPTER 32  SpanApp Service  32-1
  Information About SpanApp Service  32-1
  addDestinationPortsToSpanSession  32-1
  addFilteredVlanRangeToSpanSession  32-2
  addFilteredVlansToSpanSession  32-2
  addPortsToSpanPermitList  32-3
  addSourcesToSpanSession  32-3
  addSourcesToSpanSessionByRange  32-4
  createSpanSessions  32-4
  createSpanSessionsInNetworkElement  32-5
  deleteSpanSessions  32-6
  disableInterfacesMonitor  32-7
  enableInterfacesMonitor  32-7
  getLocalSpanSessionsInNetwork  32-8
  getSpanDestinationPorts  32-8
  getSpanPermitListSettingsInNetwork  32-9
  getSpanSessions  32-9
  getSpanSessionsAssociatedToVlan  32-10
  getSpanSessionsInNetwork  32-10
  getSpanSources  32-11
  modifyFilteredVlanRangeForSpanSession  32-11
  modifyFilteredVlansByRangeForSpanSession  32-12
  modifyFilteredVlansForSpanSession  32-12
  modifySpanPermitListSettingsInNetworkElement  32-13
  modifySpanSessions  32-13
  modifySpanSourceTrafficDirection  32-14
modifySpanSources
removeDestinationPortsFromSpanSession
removeFilteredVlanRangeInSpanSession
removeFilteredVlansInSpanSession
removePortsFromSpanPermitList
removeSourcesFromSpanSession
removeSourcesInSpanSession

CHAPTER 33

StpApp Service
Information About StpApp Service
bindPvstSettingToVlans
bindStpNetworkInterfaceSettingForMstInstances
bindStpNetworkInterfaceSettingForVlans
bindStpNetworkInterfaceSettingsToNetworkInterface
bindVlansToMstInstance
createMstInstance
createMstInstanceByRange
createMstInstances
createMstInstancesInNetworkElement
deleteMstInstances
getBlockingStpPortsCountInMsts
getBlockingStpPortsCountInVlans
getForwardingStpPortsCountInMsts
getForwardingStpPortsCountInVlans
getMstGlobalSettings
getMstIdsInNetwork
getMstInstances
getMstInstancesInNetwork
getNetworkInterfacesWithStpSettings
getOperationalStpSettingForVlans
getStpDisabledVlansInNetwork
getStpEnabledVlansInNetwork
getStpGlobalSettings
getStpMode
getStpNetworkInterfaceSettings
getStpNetworkInterfaceSettingsForAccessPortsInMst
getStpNetworkInterfaceSettingsForPortsInVlan 33-14
getStpNetworkInterfaceSettingsForPortsInVlansByInterfaces 33-15
getStpNetworkInterfaceSettingsForTrunkPortsInMst 33-15
getStpNetworkInterfaceSettingsForVlanMemberPortsInMst 33-16
getStpSettingsForMst 33-16
getStpSettingsForVlan 33-17
getStpStatusForNetworkInterfacesInVlan 33-17
getTotalStpPortsCountInMsts 33-18
getTotalStpPortsCountInVlans 33-18
getVlansWithPvrstSetting 33-18
modifyMstGlobalSettings 33-19
modifyMstInstances 33-19
modifyPvstSettings 33-20
modifyStpGlobalSettings 33-21
modifyStpNetworkInterfaceSettingForMst 33-21
modifyStpNetworkInterfaceSettingForVlan 33-21
modifyStpNetworkInterfaceSettings 33-22
modifyStpSettingForNetworkInterfaces 33-22
modifyStpType 33-23
unbindVlansFromMstInstance 33-23

CHAPTER 34

TrafficStormControlApp Service 34-1
Information About TrafficStormControlApp Service 34-1
disableBroadcastStormControlInInterfaces 34-1
disableMulticastStormControlInInterfaces 34-2
disableStormControlInInterfaces 34-2
disableUnicastStormControlInInterfaces 34-3
enableBroadcastStormControlInInterfaces 34-3
enableMulticastStormControlInInterfaces 34-4
enableUnicastStormControlInInterfaces 34-4
getInterfacesWithTrafficStormControlSettingInNetworkElement 34-5
getTrafficStormControlDisabledInterfacesInNetworkElement 34-5
getTrafficStormControlSettingInInterfaces 34-6
getTrafficStormControlledInterfacesInNetworkElement 34-6
modifyStormControlInInterfaces 34-7
TunnelApp Service  35-1

Information About TunnelApp Service  35-1
bindKeepAliveSettingToNetworkInterfaces  35-1
bindPathMtuDiscoverySettingToNetworkInterfaces  35-2
createIpGreTunnelNetworkInterfaces  35-2
deleteIpGreTunnelNetworkInterfaces  35-2
disableTunnelService  35-3
enableTunnelService  35-3
getAllTunnelInterfacesInNetworkElement  35-4
getTunnelDestinationInterface  35-4
getTunnelGlobalSetting  35-4
getTunnelNetworkInterfaces  35-5
getTunnelSourceNetworkInterface  35-5
modifyIpGreTunnelNetworkInterfaces  35-5

UserApp Service  36-1

Information About UserApp Service  36-1
bindUserToAllNetworkElements  36-1
bindUserToAllNetworkGroups  36-1
bindUserToNetworkElements  36-1
bindUserToNetworkGroups  36-2
createUsers  36-2
deleteUsers  36-2
getAllUsers  36-3
getDefaultNetworkElementUserCredentials  36-3
getNetworkElementGroupsForUser  36-3
getNetworkElementsForUser  36-3
getPreferences  36-4
getUserCredentialsForNetworkElementGroups  36-4
getUserCredentialsForNetworkElements  36-4
getUsers  36-4
modifyDefaultNetworkElementUserCredentials  36-5
modifyPreferences  36-5
modifyUserCredentialsForNetworkElementGroups  36-5
modifyUserCredentialsForNetworkElements  36-5
modifyUsers 36-6
unbindUserFromAllNetworkElements 36-6
unbindUserFromAllNetworkGroups 36-6
unbindUserFromNetworkElements 36-7
unbindUserFromNetworkGroups 36-7

CHAPTER 37  
VdcApp Service 37-1
Information About VdcApp Service 37-1
bindPortsToVdcs 37-1
bindVdcsWithResourceTemplate 37-2
createResourceTemplates 37-2
createVdcs 37-3
createVdcsWithResourceTemplate 37-3
deleteResourceTemplates 37-4
deleteVdcs 37-4
getAllVdcsWithChassis 37-4
getResourceTemplatesInNetworkElement 37-5
getVdcs 37-5
getVdcsInChassis 37-6
modifyResourceTemplates 37-6
modifyVdcs 37-6
unbindVdcsWithResourceTemplate 37-7

CHAPTER 38  
VlanApp Service 38-1
Information About VlanApp Service 38-1
addAllowedVlansOnTrunks 38-1
bindAccessPortsToVlan 38-2
bindHostPortsToPrivateVlan 38-2
bindNativeVlanOnTrunks 38-3
bindPromiscuousPortsToPrivateVlan 38-3
bindSecondaryVlansToPrimary 38-4
bindSecondaryVlansToVlanNetworkInterface 38-4
createSecondaryVlans 38-5
createVlanInNetwork 38-5
createVlans 38-6
deleteAllowedVlansOnTrunks 38-7
deleteVlans 38-7
deleteVlansInNetwork 38-8
disablePrivateVlanService 38-8
enablePrivateVlanService 38-9
getActivePortsCountInVlans 38-9
getInactivePortsCountInVlans 38-10
getInternalVlansInNetworkElement 38-10
getPortsCountInVlans 38-11
getPortsInVlan 38-11
getPrivateVlanServiceState 38-12
getPrivateVlansInNetworkElement 38-12
getSecondaryVlans 38-13
getTrunksForVlan 38-13
getVlanGlobalSettings 38-14
getVlanIdsInNetwork 38-14
getVlanNetworkInterfaces 38-14
getVlans 38-15
getVlansInNetwork 38-15
getVlansInNetworkElement 38-16
getVlansWithSecurityAssociations 38-17
getVlansWithSpanAssociations 38-17
getVtpGlobalSettings 38-18
modifyAllowedVlansOnTrunks 38-18
modifySecondaryVlansForPromiscuousPorts 38-19
modifyVlanGlobalSetting 38-19
modifyVlanTypeToNormal 38-20
modifyVlanTypeToPrimary 38-20
modifyVlanTypeToSecondary 38-21
modifyVlans 38-21
unbindAccessPortsFromVlan 38-22
unbindHostPortsFromPrivateVlan 38-22
unbindNativeVlanOnTrunks 38-23
unbindPromiscuousPortsFromPrivateVlan 38-23
unbindSecondaryVlansFromPrimary 38-24
unbindSecondaryVlansFromVlanNetworkInterface 38-24
## Chapter 39: VpcApp Service

### Information About VpcApp Service

- `createVpc`
- `createVpcDomains`
- `createVpcDomainsForIds`
- `createVpcForSpecifiedEndPoints`
- `deleteVpcDomains`
- `deleteVpcs`
- `deleteVpcsForVpcIds`
- `disableVpc`
- `disableVpcForPortChannels`
- `disableVpcPeerLinkForPortChannels`
- `enableVpc`
- `enableVpcPeerLinkForPortChannels`
- `getAllVpcs`
- `getMultiChassisPortChannelsInNetworkElements`
- `getVpcDomainsInNetworkElements`
- `getVpcldsInNetworkElements`
- `getVpcPeerLinkEndPoints`
- `getVpcStateOfNetworkElements`
- `getVpcs`
- `getVpcsForIds`
- `getVpcsInNetworkElements`
- `modifyVpcDomains`
- `modifyVpcIdForPortChannel`
- `modifyVpcs`
- `synchronizeGlobalSettingsForPrimaryAndSecondary`
- `synchronizePrimaryAndSecondary`

## Chapter 40: VrfApp Service

### Information About VrfApp Service

- `addIpNetworkInterfaces`
- `addIpNetworkInterfacesToDefaultVrf`
- `addIpv4StaticRoutes`
- `addIpv6StaticRoutes`
createVrfs  40-2
deleteVrfs  40-3
gGetAllNetworkElements  40-3
gGetIpNetworkInterfaces  40-3
gGetMulticastRoutingInstancesForVrf  40-3
gGetUnicastRoutingInstancesForVrf  40-4
gGetVrfsInNetworkElement  40-4
removeStaticRoutes  40-4
Preface

This preface describes the audience, organization, and conventions of the *Cisco DCNM Web Services API Guide, Release 4.2*. It also provides information on how to obtain related documentation.

This chapter includes the following sections:

- Audience, page xxxiii
- Organization, page xxxiii
- Document Conventions, page xxxiv
- Related Documentation, page xxxv
- Obtaining Documentation and Submitting a Service Request, page xxxv

Audience

This publication is for experienced users who configure and maintain NX-OS devices.

Organization

This reference is organized as follows:

<table>
<thead>
<tr>
<th>Chapter and Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1, “Overview”</td>
<td>Describes the Cisco NX-OS Web Services API.</td>
</tr>
<tr>
<td>Chapter 2, “AaaApp Service”</td>
<td>Describes the AAA Service API.</td>
</tr>
<tr>
<td>Chapter 3, “AclApp Service”</td>
<td>Describes the ACL Service API.</td>
</tr>
<tr>
<td>Chapter 4, “CollectorApp Service”</td>
<td>Describes the Collector Service API.</td>
</tr>
<tr>
<td>Chapter 5, “CollectorExtendedApp Service”</td>
<td>Describes the Collector Extended Service API.</td>
</tr>
<tr>
<td>Chapter 6, “ConfigArchiveApp Service”</td>
<td>Describes the Configuration Archive Service API.</td>
</tr>
<tr>
<td>Chapter 7, “DaiApp Service”</td>
<td>Describes the DAI Service API.</td>
</tr>
<tr>
<td>Chapter 8, “DataPurgingApp Service”</td>
<td>Describes the Data Purging Service API.</td>
</tr>
<tr>
<td>Chapter 9, “DeviceListApp Service”</td>
<td>Describes the Device List Service API.</td>
</tr>
<tr>
<td>Chapter 10, “DhcpSnoopingApp Service”</td>
<td>Describes the DHCP Snooping Service API.</td>
</tr>
<tr>
<td>Chapter 11, “Dot1xApp Service”</td>
<td>Describes the 802.1X Service API.</td>
</tr>
</tbody>
</table>
Preface

Document Conventions

This document uses the following conventions:

**Note**

Means reader *take note*. Notes contain helpful suggestions or references to material not covered in the manual.
Related Documentation

Cisco DCNM documentation is available at the following URL:
The documentation set for Cisco DCNM includes the following documents:

Release Notes

Cisco DCNM Release Notes, Release 4.2

DCNM Configuration Guides

Cisco DCNM Getting Started with Virtual Device Contexts, Release 4.2
Cisco DCNM Fundamentals Configuration Guide, Release 4.2
Cisco DCNM Interfaces Configuration Guide, Release 4.2
Cisco DCNM Layer 2 Switching Configuration Guide, Release 4.2
Cisco DCNM Web Services API Guide, Release 4.2
Cisco DCNM Security Configuration Guide, Release 4.2
Cisco DCNM Unicast Routing Configuration Guide, Release 4.2
Cisco DCNM Virtual Device Context Configuration Guide, Release 4.2
Cisco DCNM Software Upgrade Guide, Release 4.2

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional
information, see the monthly What’s New in Cisco Product Documentation, which also lists all new and
revised Cisco technical documentation, at:

Subscribe to the What’s New in Cisco Product Documentation as a Really Simple Syndication (RSS) feed
and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free
service and Cisco currently supports RSS Version 2.0.
Overview

This chapter describes the Data Center Network Management (DCNM) web services’ application program interface (API).

This chapter includes the following sections:

- Information About Web Services, page 1-1
- Licensing Requirements for Web Services, page 1-6
- Additional References, page 1-6

Information About Web Services

You can configure NX-OS devices by using the Cisco DCNM web services API on the DCNM server. You create XML-based API requests by using the SOAP protocol. The DCNM server configures devices using the XML management interface. For more information about the XML management interface, see the Cisco Nexus 7000 Series NX-OS XML Management Interface User Guide, Release 4.2.

The web services API provides a subset of the features available in the DCNM client. For information about DCNM, see the Cisco DCNM Fundamentals Configuration Guide, Release 4.2.

This section includes the following topics:

- SOAP Messages, page 1-1
- Error Handling, page 1-6

SOAP Messages

The SOAP messages that you exchange with the DCNM server contain a user ID and password in the header and an API call in the body.

You can obtain the XSD and WSDL files that describe the DCNM web services on the DCNM server in the ws-client folder of the DCNM installation directory. By default, the DCNM server is installed at C:\Program Files\Cisco Systems\DCNM.

For more information about SOAP, see http://www.w3.org/TR/soap.

This section includes the following topics:

- SOAP Message to Create Session ID, page 1-2
- SOAP Message to Make API Call, page 1-2
**Send document comments to nexus7k-docfeedback@cisco.com**

**Chapter 1      Overview**

**Information About Web Services**

**SOAP Message to Create Session ID**

Example 1-1 shows the initial SOAP message that you send to obtain a session ID with the createSessionID method in the SecurityApp service. All subsequent SOAP messages use the returned session ID. You use the createSessionID method in the Security AppService and specify the user ID and password of the DCNM server.

**Example 1-1   SOAP Message to Obtain Session ID**

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header>
    <util:credentialHeader xmlns:util="http://util.common.dcm.dcbu.cisco.com"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://util.common.dcm.dcbu.cisco.com util.xsd">
      <userId>administration</userId>
      <password>administration</password>
    </util:credentialHeader>
  </env:Header>
  <env:Body>
    <m:createSessionID
      xmlns:m="http://security.app.ws.dcm.dcbu.cisco.com/SecurityAppService">
    </m:createSessionID>
  </env:Body>
</env:Envelope>
```

Example 1-2 shows the returned session ID “FFEE034DDFAFASEREW8383FDSFSD” that you use in subsequent API calls.

**Example 1-2   Response to SOAP Message to Obtain Session ID**

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <env:Body>
    <m:createSessionIDResponse
      xmlns:m="http://security.app.ws.dcm.dcbu.cisco.com/SecurityAppService">
      <return xmlns:basictypes="http://basictypes.dcm.dcbu.cisco.com"
        xmlns:common="http://common.dcm.dcbu.cisco.com"
        xmlns:datatype="http://datatype.model.dcm.dcbu.cisco.com"
        xmlns:enumkind="http://enumkind.model.dcm.dcbu.cisco.com"
        xmlns:util="http://util.common.dcm.dcbu.cisco.com"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.w3.org/2001/XMLSchema xml.xsd"
        xsi:type="xs:string">FFEE034DDFAFASEREW8383FDSFSD</return>
    </m:createSessionIDResponse>
  </env:Body>
</env:Envelope>
```

**SOAP Message to Make API Call**

Example 1-3 shows the SOAP message that you send to request the chassis inventory with method getAllChassisInNetwork in the InventoryApp service.

**Example 1-3   SOAP Message to Request Chassis Inventory**

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
```

```xml
```
```
Example 1-4 shows the response to the request for chassis inventory.

**Example 1-4  Response to SOAP Message to Request Chassis Inventory**

```xml
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:xs="http://www.w3.org/2001/XMLSchema"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <env:Header/>
  <env:Body>
    <m:getAllChassisInNetworkResponse xmlns:m="http://inventory.app.ws.dcm.dcbu.cisco.com/InventoryAppService">
      <return xmlns:basictypes="http://basictypes.dcm.dcbu.cisco.com"
              xmlns:common="http://common.dcm.dcbu.cisco.com"
              xmlns:datatype="http://datatype.model.dcm.dcbu.cisco.com"
              xmlns:enumkind="http://enumkind.model.dcm.dcbu.cisco.com"
              xmlns:physical="http://physical.model.dcm.dcbu.cisco.com"
              xmlns:tns="http://physical.model.dcm.dcbu.cisco.com"
              xmlns:util="http://util.common.dcm.dcbu.cisco.com"
              xmlns:vdc="http://vdc.model.dcm.dcbu.cisco.com"
              xsi:schemaLocation="http://physical.model.dcm.dcbu.cisco.com physical.xsd" xsi:type="physical:Chassis_ListType">
        <itemList xsi:type="physical:Chassis">
          <instClassName>com.cisco.dcbu.dcm.model.physical.Chassis</instClassName>
          <instanceClassId>6143</instanceClassId>
          <instanceName>com.cisco.dcbu.dcm.model.physical.Chassis:neId=2,chassisId=TBM11256510</instanceName>
          <instanceState>Unmodified</instanceState>
          <version>3</version>
          <description>Nexus7000 C7010 (10 Slot) Chassis</description>
          <name>Chassis</name>
          <productId>N7K-C7010</productId>
          <serialNumber>TBM11256510</serialNumber>
          <chassisPowerSupplyRef xsi:type="physical:ChassisPowerSupplySetting">
            <instClassName>com.cisco.dcbu.dcm.model.physical.ChassisPowerSupplySetting</instClassName>
            <powerSupplyRedundancyMode>Redundant</powerSupplyRedundancyMode>
          </chassisPowerSupplyRef>
        </itemList>
        <itemList xsi:type="physical:Chassis">
          <instClassName>com.cisco.dcbu.dcm.model.physical.ChassisPowerSupplyStatus</instClassName>
          <instanceClassId>6252</instanceClassId>
          <instanceName>com.cisco.dcbu.dcm.model.physical.ChassisPowerSupplyStatus:neId=2,chassisId=TBM11256510</instanceName>
          <instanceState>Unmodified</instanceState>
          <version>0</version>
        </itemList>
      </return>
    </m:getAllChassisInNetworkResponse>
  </env:Body>
</env:Envelope>
```
Information About Web Services

Send document comments to nexus7k-docfeedback@cisco.com

<availablePowerSupply>2329.00</availablePowerSupply>
<operationalRedundancyStatus>Redundant</operationalRedundancyStatus>
<totalPowerSupply>5480.00</totalPowerSupply>
<usedPowerSupply>3151.0</usedPowerSupply>
</chassisPowerSupplyStatusRef>

<networkElementRef xsi:type="vdc:NetworkElement">
<instClassName>com.cisco.dcbu.dcm.model.vdc.NetworkElement</instClassName>
<instanceClassId>2</instanceClassId>
<instanceName>com.cisco.dcbu.dcm.model.vdc.NetworkElement:neId=2</instanceName>
<instanceState>Unmodified</instanceState>
<version>6</version>
</networkElementRef>

<abstractNetworkElementStatusRef xsi:type="vdc:AbstractNetworkElementStatus">
<instClassName>com.cisco.dcbu.dcm.model.vdc.AbstractNetworkElementStatus</instClassName>
<instanceClassId>6146</instanceClassId>
<instanceName>com.cisco.dcbu.dcm.model.vdc.AbstractNetworkElementStatus:neId=2</instanceName>
<instanceState>Unmodified</instanceState>
</abstractNetworkElementStatusRef>

<hostName>DCNM-9</hostName>
<mgmtIpAddress>172.27.139.76</mgmtIpAddress>
<platformName>Nexus7000</platformName>
<platformType>Cisco Nexus7000 Series</platformType>
<platformVersion>4.0(1)</platformVersion>
</networkElementRef>

<redundancySettingsRef xsi:type="physical:RedundancySetting">
<instClassName>com.cisco.dcbu.dcm.model.physical.RedundancySetting</instClassName>
<redundancyMode>HA</redundancyMode>
</redundancySettingsRef>

<redundancyStatusRef xsi:type="physical:RedundancyStatus">
<instClassName>com.cisco.dcbu.dcm.model.physical.RedundancyStatus</instClassName>
<instanceClassId>6250</instanceClassId>
<instanceName>com.cisco.dcbu.dcm.model.physical.RedundancyStatus:neId=2,chassisId=TBM11256510</instanceName>
<instanceState>Unmodified</instanceState>
</redundancyStatusRef>

<slotCol collectionType="Set" xsi:type="physical:Slot_SetType">
<itemSet xsi:type="physical:Slot">
<instClassName>com.cisco.dcbu.dcm.model.physical.Slot</instClassName>
<instanceClassId>6150</instanceClassId>
<instanceName>com.cisco.dcbu.dcm.model.physical.Slot:neId=2,chassisId=TBM11256510,slotNumber=3</instanceName>
<instanceState>Unmodified</instanceState>
</itemSet>
</slotCol>
Information About Web Services

Send document comments to nexus7k-docfeedback@cisco.com

<itemSet xsi:type="physical:Slot">
  <instClassName>com.cisco.dcbu.dcm.model.physical.Slot</instClassName>
  <instanceClassId>6148</instanceClassId>
  <instanceName>com.cisco.dcbu.dcm.model.physical.Slot:neId=2,chassisId=TBM11256510,slotNumber=1</instanceName>
  <instanceState>Unmodified</instanceState>
  <version>1</version>
  <chassisRef xsi:type="physical:Chassis">
    <instanceName>com.cisco.dcbu.dcm.model.physical.Chassis:neId=2,chassisId=TBM11256510</instanceName>
    <slotNumber>1</slotNumber>
  </chassisRef>
</itemSet>

<itemSet xsi:type="physical:Slot">
  <instClassName>com.cisco.dcbu.dcm.model.physical.Slot</instClassName>
  <instanceClassId>6152</instanceClassId>
  <instanceName>com.cisco.dcbu.dcm.model.physical.Slot:neId=2,chassisId=TBM11256510,slotNumber=5</instanceName>
  <instanceState>Unmodified</instanceState>
  <version>1</version>
  <chassisRef xsi:type="physical:Chassis">
    <instanceName>com.cisco.dcbu.dcm.model.physical.Chassis:neId=2,chassisId=TBM11256510</instanceName>
    <slotNumber>5</slotNumber>
  </chassisRef>
</itemSet>

<itemSet xsi:type="physical:Slot">
  <instClassName>com.cisco.dcbu.dcm.model.physical.Slot</instClassName>
  <instanceClassId>6156</instanceClassId>
  <instanceName>com.cisco.dcbu.dcm.model.physical.Slot:neId=2,chassisId=TBM11256510,slotNumber=12</instanceName>
  <instanceState>Unmodified</instanceState>
  <version>1</version>
  <chassisRef xsi:type="physical:Chassis">
    <instanceName>com.cisco.dcbu.dcm.model.physical.Chassis:neId=2,chassisId=TBM11256510</instanceName>
    <slotNumber>12</slotNumber>
  </chassisRef>
</itemSet>

<itemSet xsi:type="physical:Slot">
  <instClassName>com.cisco.dcbu.dcm.model.physical.Slot</instClassName>
  <instanceClassId>6154</instanceClassId>
  <instanceName>com.cisco.dcbu.dcm.model.physical.Slot:neId=2,chassisId=TBM11256510,slotNumber=11</instanceName>
  <instanceState>Unmodified</instanceState>
  <version>1</version>
  <chassisRef xsi:type="physical:Chassis">
    <instanceName>com.cisco.dcbu.dcm.model.physical.Chassis:neId=2,chassisId=TBM11256510</instanceName>
    <slotNumber>11</slotNumber>
  </chassisRef>
</itemSet>

</slotCol>
</itemList>
</return>
</m:getAllChassisInNetworkResponse>
Licensing Requirements for Web Services

The following table shows the licensing requirements for this feature:

<table>
<thead>
<tr>
<th>Product</th>
<th>License Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCNM</td>
<td>Web services requires no license. Any feature not included in a license package is bundled with the Cisco DCNM and is provided at no charge to you. For a complete explanation of the DCNM licensing scheme, see the Cisco DCNM Licensing Guide.</td>
</tr>
</tbody>
</table>

Additional References

For additional information related to implementing the Web Services API, see the following sections:

- Related Documents, page 1-6
- Standards, page 1-6

Related Documents

<table>
<thead>
<tr>
<th>Related Topic</th>
<th>Document Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCMN client</td>
<td>Cisco DCNM Fundamentals Configuration Guide, Release 4.2</td>
</tr>
</tbody>
</table>

Standards

<table>
<thead>
<tr>
<th>Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>—</td>
</tr>
</tbody>
</table>
AaaApp Service

This chapter describes the DCNM web services’ API methods for the AaaApp service.

Information About AaaApp Service

Authentication, Authorization, and Accounting (AAA) services provide the primary framework to set up access control on a router or access server. Access control allows you to control who is allowed access to a network server and what services they are allowed to access.

Authentication is the process of identifying an individual user, usually based on a username and password.

Authorization is the process of granting or denying a user access to network resources once the user has been authenticated through the username and password. The amount of information and the amount of services that the user has access to depends on the user’s authorization level.

Accounting is the process of keeping track of a user’s activity while accessing the network resources, including the amount of time spent in the network, the services accessed, and the amount of data transferred during the session.

The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Create APIs—Create new AAA servers and AAA rules.
- Modify APIs—Modify basic attributes of existing AAA servers and AAA rules
- Delete APIs—Delete existing AAA servers and AAA rules
- Enable and Disable APIs—Enable and disable AAA feature or TACACS in the device

This chapter contains APIs for the following features:

- AAA Rules
- AAA Server Groups

bindNetworkInterfaceToAaaServerGroups

Assigns a network interface to one or more AAA server groups as source interface. If server group has an existing source interface association, this new interface association will overwrite the existing association.
bindNetworkInterfaceToGlobalRadiusServerSetting

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameId or aaaServerGrpInstanceNameIdColis null
- If aaaServerGrpInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type AaaServerGroup InstanceNameId
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the given server type is TACACS+ and TACACS+ is disabled in the given network element that is running Cisco NX-OS.
- If network interface and AAA server groups are not from the same network element.

IntegrityException is thrown if any of the following situation occurs:
- If the object for the given networkInterfaceInstanceNameId doesn't exist in the device.
- If the objects for the given aaaServerGrpInstanceNameIdCol doesn't exist in the device.

Parameters
opContext—Operational context
aaaServerGrpInstanceNameIdCol—list of InstanceNameId of the AaaServerGroup for which the source interface needs to be assigned
networkInterfaceInstanceNameId—InstanceNameId of the network interface.

Return Value
void

bindNetworkInterfaceToGlobalRadiusServerSetting

Assigns a network interface as the source interface for the globally configured RADIUS servers in a network element. If global RADIUS server has an existing source interface association, this new interface association will overwrite the existing association.

ValidationException is thrown if any of the following situation occurs:
- If networkInterfaceInstanceNameIdis null
- If aaaServerGrpInstanceNameIdCol is not type AaaServerGroup InstanceNameId
- If AAA is disabled in the given network element that is running Cisco IOS.

IntegrityException is thrown if any of the following situation occurs:
- If the object for the given networkInterfaceInstanceNameId doesn't exist in the device.

Parameters
opContext—Operational context
networkInterfaceInstanceNameId—InstanceNameId of the network interface.

Return Value
void
bindNetworkInterfaceToGlobalTACACSServerSetting

Assigns a network interface as the source interface for the globally configured TACACS server settings in a network element. If global TACACS server setting has an existing source interface association, this new interface association will overwrite the existing association.

ValidationException is thrown if any of the following situation occurs:

- If networkInterfaceInstanceNameId is null
- If aaaServerGrpInstanceNameIdCol is not AaaServerGroup InstanceNameId
- If AAA is disabled in the given network element that is running Cisco IOS.
- If TACACS is disabled in the given network element that is running Cisco NX-OS.

Parameters

- opContext—Operational context
- networkInterfaceInstanceNameId—InstanceNameId of the network interface.

Return Value

- void

createAaaServerGroups

Creates one or more AAA server groups and its associated Group AAA servers in a network element. If any Group server that is not configured as a global server, it will create Global servers respected to group servers.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameId is null or it is not a valid network element InstanceNameId.
- If the aaaServerGrpCol is null.
- If the aaaServerGrpCol contains one or more null element, or the collection is empty
- If aaaServerGrpCol collection contains an element that is not of type AaaServerGroup.
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the given AAA server Group is TACACS+ server and TACACS+ is disabled in the given network element that is running Cisco NX-OS.

PropertiesException is thrown if any of the following situation occurs:

- If AAA Server Group is not valid.

Example:

- Server Group name is null

IntegrityException is thrown if any of the following situation occurs:

- If the Server Group with the same name already exist in the device.
createAccountingRulesInNetworkElement

**Parameters**

- **opContext**—Operational context
- **neInstanceNameId**—InstanceNameId of the Network Element where the AAA Server Groups needs to be created.
- **aaaServerGrpCol**—List of AaaServerGroup objects that needs to be created.

**Return Value**

The List of InstanceNameId of the AaaServerGroup objects that gets created by this method.

---

createAccountingRulesInNetworkElement

Creates one or more accounting rules in a network element.

**ValidationException** is thrown if any of the following situation occurs:

- If AAA is disabled in the given network element that is running Cisco IOS.
- If the acRuleCol is null.
- If the acRuleCol contains one or more null element, or the collection is empty.
- If the neInstanceNameId is null or it is not a valid network element InstanceNameId.
- If any of the given rule doesn't have a method.
- If the device is running Cisco IOS, and any of the given rules has more than 4 methods.
- If the given network element is running Cisco NX-OS, and any of the given rules has more than 10 methods.

**PropertiesException** is thrown if any of the following situation occurs:

- If Accounting Rule is not valid.

**Example:**

- Rule name is null

**IntegrityException** is thrown if any of the following situation occurs:

- If the accounting rule with the same name and service already exist in the device.
- If a method uses server groups, then those server group references should be available. Also, that server group must be present in the database.

**Parameters**

- **opContext**—Operational context
- **neInstanceNameId**—InstanceNameId of the Network Element in where the accounting rules needs to be created.
- **acRuleCol**—List of AccountingRule objects that needs to be created.

**Return Value**

The List of InstanceNameId of the AccountingRule objects that gets created by this method.
createAuthenticationRulesInNetworkElement

Creates one or more authentication rules in a network element.

ValidationException is thrown if any of the following situation occurs:

- If AAA is disabled in the given network element that is running Cisco IOS.
- If the atRuleCol is null.
- If the atRuleCol contains one or more null element, or the collection is empty
- If the neInstanceNameId is null or it is not a valid network element InstanceNameId.
- If any of the given rule doesn't have a method
- If the given network element is running Cisco IOS, and any of the given rules has more than 4 methods
- If the given network element is running Cisco NX-OS and any of the given rules has more than 10 methods

PropertiesException is thrown if any of the following situation occurs:

- If Authentication Rule is not valid.

Example:

- Rule name is null

IntegrityException is thrown if any of the following situation occurs:

- If the authentication rule with the same name and service already exist in the device.
- If a method uses server groups, then those server group references should be available. Also, that server group must be present in the database.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the Network Element where the authentication rules needs to be created.
- atRuleCol—List of AuthenticationRule objects that needs to be created.

Return Value

The List of InstanceNameId of the AuthenticationRule objects that gets created by this method.

createAuthorizationRulesInNetworkElement

Creates one or more authorization rules in a network element.

ValidationException is thrown if any of the following situation occurs:

- If AAA is disabled in the given network element that is running Cisco IOS.
- If the given network element is running Cisco NX-OS.
- If the arRuleCol is null.
- If the arRuleCol contains one or more null element, or the collection is empty
- If the neInstanceNameId is null or it is not a valid network element InstanceNameId.

PropertiesException is thrown if any of the following situation occurs:

- If Authorization Rule is not valid.

Example:

- Rule name is null

IntegrityException is thrown if any of the following situation occurs:

- If the authentication rule with the same name and service already exist in the device.

- If a method uses server groups, then those server group references should be available. Also, that server group must be present in the database.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the Network Element where the authorization rules needs to be created.
- arRuleCol—List of AuthorizationRule objects that needs to be created.

Return Value

The List of InstanceNameId of the AuthorizationRule objects that gets created by this method.
createGlobalAaaServers

Creates one or more AAA servers globally in a network element.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameId is null or it is not a valid network element InstanceNameId.
- If the aaaServerCol is null.
- If the aaaServerCol contains one or more null element, or the collection is empty
- If aaaServerCol collection contains an element that is not of type GlobalAaaServer.
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the given AAA server is TACACS+ server and TACACS+ is disabled in the given network element that is running Cisco NX-OS.

PropertiesException is thrown if any of the following situation occurs:

- If Global AAA Server is not valid.

Example:

- Server host name and Ip Address is null

IntegrityException is thrown if any of the following situation occurs:

- If the AAA server with the same name, authentication port and accounting port already exist in the device globally.
deleteAaaServerGroups

Deletes one or more AAA Server groups from a network element.

ValidationException is thrown if any of the following situation occurs:

- If AAA is disabled in the given network element that is running Cisco IOS.
- If the aaaServerGrpInstanceNameIdCol is null.
- If the aaaServerGrpInstanceNameIdCol contains one or more null element, or the collection is empty.
- If aaaServerGrpInstanceNameIdCol collection contains an element that is not of type AaaServerGroup InstanceNameId.
- If any of the given AAA server group is associated with AAA rules.

IntegrityException is thrown if any of the following situation occurs:

- If the given AaaServerGroup doesn't exist in the device.

Parameters

- **opContext**—Operational context
- **aaaServerGrpInstanceNameIdCol**—Instance name ID of one or more AaaServerGroup objects that needs to be deleted.

Return Value

void

deleteAccountingRules

Deletes one or more accounting rules from the network element.

ValidationException is thrown if any of the following situation occurs:

- If AAA is disabled in the given network element that is running Cisco IOS.
- If the acRuleInstanceNameIdCol is null.
- If the acRuleInstanceNameIdCol contains one or more null element, or the collection is empty.
- If acRuleInstanceNameIdCol collection contains an element that is not of type AccountingRule InstanceNameId.

IntegrityException is thrown if any of the following situation occurs:

Parameters

- **opContext**—Operational context
- **acRuleInstanceNameIdCol**—Instance name ID of one or more AccountingRule objects that needs to be deleted.

Return Value

void
deleteAuthenticationRules

Deletes one or more authentication rules from the network element.
ValidationException is thrown if any of the following situation occurs:
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the atRuleInstanceNameIdCol is null.
- If the atRuleInstanceNameIdCol contains one or more null element, or the collection is empty
- If atRuleInstanceNameIdCol collection contains an element that is not of type AuthenticationRule
  InstanceNameId.
IntegrityException is thrown if any of the following situation occurs:
- If the given authentication rule doesn't exist in the device.

Parameters
opContext—Operational context
atRuleInstanceNameIdCol—Instance name ID of one or more AuthenticationRule objects that needs to be deleted.

Return Value
void

deleteAuthorizationRules

Deletes one or more authorization rules from the network element.
ValidationException is thrown if any of the following situation occurs:
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the arRuleInstanceNameIdCol is null.
- If the arRuleInstanceNameIdCol contains one or more null element, or the collection is empty
- If arRuleInstanceNameIdCol collection contains an element that is not of type AuthorizationRule
  InstanceNameId.
IntegrityException is thrown if any of the following situation occurs:
- If the given authorization rule doesn't exist in the device.

Parameters
opContext—Operational context
arRuleInstanceNameIdCol—Instance name ID of one or more AuthorizationRule objects that needs to be deleted.

Return Value
void
deleteGlobalAaaServers

Deletes one or more Global AAA Servers from a network element.
ValidationException is thrown if any of the following situation occurs:
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the aaaServerInstanceNameIdCol is null.
- If the aaaServerInstanceNameIdCol contains one or more null element, or the collection is empty
- If aaaServerInstanceNameIdCol collection contains an element that is not of type GlobalAaaServer InstanceNameId.
- If group servers exist with the same IpAddress/HostName of any of the given global servers
IntegrityException is thrown if any of the following situation occurs:
- If the given GlobalAaaServer doesn't exist in the device.

Parameters
opContext—Operational context
aaaServerInstanceNameIdCol—Instance name ID of one or more GlobalAaaServer objects that needs to be deleted.

Return Value
void

disableAaa

Disables AAA on one or more network element. This API is applicable only for devices running Cisco IOS, and not applicable for Cisco NX-OS.
ValidationException is thrown if any of the following situation occurs:
- If the neInstanceNameIdCol is null.
- If the neInstanceNameIdCol contains one or more null element, or the collection is empty or it is not type InstanceNameId.
- If the given network element is running Cisco NX-OS.

Parameters
opContext—Operational context
**disableTACACS**

Disables TACACS+ on one or more network element. This API is applicable only for Cisco NX-OS and not for devices running Cisco IOS.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameIdCol is null.
- If the neInstanceNameIdCol contains one or more null element, or the collection is empty or it is not type InstanceNameId.
- If the given network element is running Cisco IOS.

**Parameters**

- **opContext**—Operational context
- **neInstanceNameIdCol**—list of InstanceNameId of the Network Element for which TACACS+ should be disabled

**Return Value**

void

**enableAaa**

Enables AAA on one or more network elements. Given the instance name ID of one or more network elements, AAA will be enabled on those elements. This API can be used to enable AAA for Cisco IOS devices. This API is not applicable for Cisco NX-OS.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameIdCol is null.
- If the neInstanceNameIdCol contains one or more null element, or the collection is empty or it is not type InstanceNameId.
- If the given network element is running Cisco NX-OS.

**Parameters**

- **opContext**—Operational context
- **neInstanceNameIdCol**—list of InstanceNameId of the Network Element for which the AAA should be enabled

**Return Value**

void
enableTACACS

Enables TACACS+ on one or more network elements. Given the instance name ID of one or more network elements, TACACS+ will be enabled on those elements. This API is only applicable for Cisco NX-OS, and not for Cisco IOS.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameIdCol is null.
- If the neInstanceNameIdCol contains one or more null element, or the collection is empty or it is not type InstanceNameId.
- If the given network element is running Cisco IOS.

Parameters

opContext—Operational context
neInstanceNameIdCol—list of InstanceNameId of the Network Element for which TACACS+ should be enabled

Return Value

void

getAaaServerGroups

Returns one or more AAA server groups. Given the instance ID of one or more AAA server groups, returns the corresponding AAA server group objects.

ValidationException is thrown if any of the following situation occurs:

- If grpInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId
- If the argument passed is null or it is not a valid AAA server group InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

opContext—Operational context
grpInstanceNameIdCol—InstanceNameId of the one or more AAA server group objects.

Return Value

The AAA server groups for the given AAA server group InstanceNameIds. The returned list will contain the list of AaaServerGroup instances.

Following associations will be there for an AAA server groups:

- Group AAA servers association.
- If the server group has any source interface association, then it will be available. But if the network interface has some other associations, then all those associations will be cleared.
- All other associations will be cleared.
getAaaServerGroupsInNetworkElement

Returns AAA server groups that are configured in a network element. Given the instance name ID of a network element, returns a list of AAA server groups.

ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null or it is not a valid network element InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId of the Network Element for which the AAA server groups are required

Return Value
The AAA server groups present in the given network element. The returned list will contain the list of AaaServerGroup instances.

Following associations will be there for an AAA server groups:

- Group AAA servers association.
- If the server group has any source interface association, then it will be available. But if the network interface has some other associations, then all those associations will be cleared.
- All other associations will be cleared.

getAaaStateOfNetworkElements

Returns state of AAA whether AAA is enabled or disabled in a list of network elements. Given the list of instance name IDs of the network elements, returns the list of Boolean values.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId.
- if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
neInstanceNameIdCol—InstanceNameId of the one or more Network Element for which the AAA state is required

Return Value
The returned list will contain Boolean instances.

Boolean value TRUE represents AAA is enabled in the given network element.
Boolean value FALSE represents AAA is disabled in the given network element.
getAccountingRules

Returns one or more accounting rules. Given the instance ID of one or more accounting rules, returns the corresponding accounting rule objects.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId
- If the argument passed is null or it is not a valid accounting rule InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

opContext—Operational context

acRuleInstanceNameIdCol—InstanceNameId of the one or more accounting rule objects.

Return Value

The accounting rules for the given accounting rule InstanceNameIds. The returned list will contain the list of AccountingRule instances.

Following associations will be there for an accounting rule:

- accounting methods (AccountingMethod)
- If accounting methods uses server groups, then those server group references will be available. But in that server group, all its references will be cleared.

getAccountingRulesInNetworkElement

Returns accounting rules configured in a network element. Given the instance ID of a network element, returns a list of accounting rules.

ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null or it is not a valid network element InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

opContext—Operational context

neInstanceNameId—InstanceNameId of the Network Element for which the accounting rules are required

Return Value

The accounting rules present in the given network element. The returned list will contain the list of AccountingRule instances.

Following associations will be there for an accounting rule:

- accounting methods(AccountingMethod)
- If accounting methods uses server groups, then those server group references will be available. But in that server group, all its references will be cleared.
getAuthenticationRules

Returns one or more authentication rules. Given the instance ID of one or more authentication rules, returns the corresponding authentication rule objects.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId
- If the argument passed is null or it is not a valid authentication rule InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

- opContext—Operational context
- atRuleInstanceNameIdCol—InstanceNameId of the one or more authentication rule objects.

Return Value

The authentication rules for the given authentication rule InstanceNameIds. The returned list will contain the list of AuthenticationRule instances.

Following associations will be there for an authentication rule:

- authentication methods(AuthenticationMethod)
- If authentication methods uses server groups, then those server group references will be available. But in that server group, all it’s references will be cleared.

getAuthenticationRulesInNetworkElement

Returns authentication rules configured in a network element. Given the instance name ID of a network element, returns a list of authentication rules.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the Network Element for which the authentication rules are required

Return Value

The authentication rules present in the given network element. The returned list will contain the list of AuthenticationRule instances.

Following associations will be there for an authentication rule:

- authentication methods(AuthenticationMethod)
- If authentication methods uses server groups, then those server group references will be available. But in that server group, all it’s references will be cleared.
getAuthorizationRules

Returns one or more authorization rules. Given the instance ID of one or more authorization rules, returns the corresponding authorization rule objects.

ValidationException is thrown if any of the following situation occurs:

- If `neInstanceNameIdCol` collection contains an element that is null or the collection is empty or it is not type `InstanceNameId`
- If the argument passed is null or it is not a valid authorization rule `InstanceNameId`.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

- `opContext`—Operational context
- `arRuleInstanceNameIdCol`—`InstanceNameId` of the one or more authorization rule objects.

Return Value

The authorization rules for the given authorization rule `InstanceNameIds`. The returned list will contain the list of `AuthorizationRule` instances.

Following associations will be there for an authorization rule:

- authorization methods (`AuthorizationMethod`)
- If authorization methods uses server groups, then those server group references will be available. But in that server group, all it's references will be cleared.

getAuthorizationRulesInNetworkElement

Returns authorization rules configured in a network element. Given the instance name ID of a network element, returns a list of authorization rules.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element `InstanceNameId`.
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the given network element is running Cisco NX-OS.

Parameters

- `opContext`—Operational context
- `neInstanceNameId`—`InstanceNameId` of the Network Element for which the authorization rules are required

Return Value

The authorization rules present in the given network element. The returned list will contain the list of `AuthorizationRule` instances.

Following associations will be there for an authorization rule:

- authorization methods (`AuthorizationMethod`)
- If authorization methods uses server groups, then those server group references will be available. But in that server group, all it's references will be cleared.
getGlobalAaaServerSettings

Returns one or more global radius/TACACS server settings. Given the instance ID of one or more global radius/TACACS server settings, returns the corresponding global radius/TACACS server settings.

ValidationException is thrown if any of the following situation occurs:

- If settingInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId
- If the argument passed is null or it is not a valid global AAA server setting InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

- opContext—Operational context
- grpInstanceNameIdCol—InstanceNameId of the one or more global AAA server objects.

Return Value

The global radius/TACACS server settings for the given AAA server InstanceNameIds. The returned list will contain the list of GlobalAaaServerSetting instances. All its associations will be cleared.

getGlobalAaaServers

Returns one or more AAA servers that are configured globally. Given the instance ID of one or more AAA servers, returns the corresponding AAA server objects.

ValidationException is thrown if any of the following situation occurs:

- If serverInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId
- If the argument passed is null or it is not a valid Global AAA server InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

- opContext—Operational context
- grpInstanceNameIdCol—InstanceNameId of the one or more global AAA server objects.

Return Value

The global AAA servers for the given AAA server InstanceNameIds. The returned list will contain the list of GlobalAaaServer instances. All its associations will be cleared.

getGlobalAaaServersForGroupAaaServers

Returns AAA servers that are configured globally respect to given group AAA servers. Given the instance name ID of a Group AAA servers, returns a list of AAA servers configured globally.
getGlobalAaaServersInNetworkElement

Returns AAA servers that are configured globally in a network element. Given the instance name ID of a network element, returns a list of AAA servers configured globally.

ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null or it is not a valid network element InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the Network Element for which the AAA servers are required

Return Value

The AAA servers configured globally in the given network element. The returned list will contain the list of GlobalAaaServer instances.

All its associations will be cleared.

getGlobalRadiusServerSettings

Returns the device level RADIUS server settings for one or more network elements. Given the instance name ID of one or more network elements, returns the corresponding global AAA server settings object for those network elements.

ValidationException is thrown if any of the following situation occurs:

- If AAA is disabled in the given network element that is running Cisco IOS.
- If neInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId

Parameters

- opContext—Operational context
getGlobalRadiusServersInNetworkElement

Returns RADIUS servers that are configured globally in a network element. Given the instance name ID of a network element, returns a list of AAA servers of type RADIUS.

ValidationException is thrown if any of the following situation occurs:
- if the argument passed is null or it is not a valid network element InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId of the Network Element for which the RADIUS servers are required

Return Value
The RADIUS servers configured globally in the given network element. The returned list will contain the list of GlobalAaaServerSetting instances of type RADIUS.

All its associations will be cleared.

getGlobalTACACSServerSettings

Returns the device level TACACS+ Server settings for one or more network elements. Given the instance name ID of one or more network elements, returns the corresponding global AAA server settings object for those network elements.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters
- opContext—Operational context
- neInstanceNameIdCol—list of InstanceNameId of the Network Element for which the TACACS+ server settings are required
Return Value

The TACACS+ server settings configured globally in the given network element. The returned list will contain the list of GlobalAaaServerSetting instances of type TACACS+.

Following associations will be there for an AAA server groups:

- Source Interface association. But if the network interface has some other associations, then all those associations will be cleared.

getGlobalTACACSServersInNetworkElement

Returns TACACS+ servers that are configured globally in a network element. Given the instance name ID of a network element, returns a list of AAA servers of type TACACS+.

ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null or it is not a valid network element InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.
- If TACACS+ is disabled in the given network element that is running Cisco NX-OS.

Parameters

opContext—Operational context
neInstanceNameId—InstanceNameId of the Network Element for which the TACACS+ servers are required

Return Value

The TACACS+ servers configured globally in the given network element. The returned list will contain the list of GlobalAaaServer instances of type TACACS+.

All its associations will be cleared.

getRadiusServerGroupsInNetworkElement

Returns AAA server groups of type RADIUS that are configured in a network element. Given the instance name ID of a network element, returns a list of AAA server groups of type RADIUS.

ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null or it is not a valid network element InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

opContext—Operational context
neInstanceNameId—InstanceNameId of the Network Element for which the RADIUS server groups are required

Return Value

The RADIUS server groups present in the given network element. The returned list will contain the list of AaaServerGroup instances of type RADIUS.

Following associations will be there for an AAA server groups:
getTACACSServerGroupsInNetworkElement

Returns AAA server groups of type TACACS+ that are configured in a network element. Given the instance name ID of a network element, returns a list of AAA server groups of type TACACS+. Following associations will be available:

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If AAA is disabled in the given network element that is running Cisco IOS.
- If TACACS+ is disabled in the given network element that is running Cisco NX-OS

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the Network Element for which the TACACS+ server groups are required

Return Value

The TACACS+ server groups present in the given network element. The returned list will contain the list of AaaServerGroup instances of type TACACS+.

Following associations will be there for an AAA server groups:

- Group AAA servers association.
- If the server group has any source interface association, then it will be available. But if the network interface has some other associations, then all those associations will be cleared.
- All other associations will be cleared.

getTACACSStateOfNetworkElements

Returns state of TACACS+ service whether TACACS+ is enabled or disabled in a list of network elements. Given the list of instance name IDs of the network elements, returns a list of Boolean values.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId.
- If the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—InstanceNameId of the one or more Network Element for which the TACACS+ state is required
modifyAaaServerGroups

Modifies one or more existing AAA Server groups in a network element.
ValidationException is thrown if any of the following situation occurs:

- If the aaaServerCol is null.
- If the aaaServerCol contains one or more null element, or the collection is empty.
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the given AAA server is TACACS+ server and TACACS+ is disabled in the given network element that is running Cisco NX-OS.

PropertiesException is thrown if any of the following situation occurs:

- If AAA server is not valid.

Example:

- Server host name and Ip Address is updated

IntegrityException is thrown if any of the following situation occurs:

- If the given AAA Server doesn't exist globally in the device.
- If the Group AAA Servers that are associated with Server Group except private servers are not configured globally.

Parameters

- opContext—Operational context
- aaaServerGrpCol—List of modified AaaServerGroup objects that will replace the existing objects.

Return Value

- void

modifyAccountingRules

Modifies one or more existing accounting rules in a network element.
ValidationException is thrown if any of the following situation occurs:

- If AAA is disabled in the given network element that is running Cisco IOS.
- If the acRuleCol is null.
- If the acRuleCol contains one or more null element, or the collection is empty.
- If any of the given rule doesn't have a method.
- If the device is running Cisco IOS, and any of the given rules has more than 4 methods.

Parameters

- acRuleCol—List of modified AccountingRule objects that will replace the existing objects.
modifyAuthenticationRules

Modifies one or more existing authentication rules in a network element.
ValidationException is thrown if any of the following situation occurs:
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the atRuleCol is null.
- If the atRuleCol contains one or more null element, or the collection is empty
- If any of the given rule doesn't have a method
- If the device is running Cisco IOS, and any of the given rules has more than 4 methods
- If the given network element is running Cisco NX-OS, and any of the given rules has more than 10 methods
PropertiesException is thrown if any of the following situation occurs:
- If Authentication Rule is not valid.
Example:
- Rule name is updated
IntegrityException is thrown if any of the following situation occurs:
- If the given authentication rule doesn't exist in the device.
- If a method uses server groups, then those server group references should be available. Also, that server group must be present in the database.

Parameters
opContext—Operational context
atRuleCol—List of modified AuthenticationRule objects that will replace the existing objects.

Return Value
void
modifyAuthorizationRules

Modifies one or more existing authorization rules in a network element.

ValidationException is thrown if any of the following situation occurs:

- If AAA is disabled in the given network element that is running Cisco IOS.
- If the given network element is running Cisco NX-OS.
- If the arRuleCol is null.
- If the arRuleCol contains one or more null element, or the collection is empty
- If any of the given rule doesn't have a method
- If the device is running Cisco IOS, and any of the given rules has more than 4 methods

PropertiesException is thrown if any of the following situation occurs:

- If Authorization Rule is not valid.

Example:

- Rule name is updated

IntegrityException is thrown if any of the following situation occurs:

- If the authorization rule doesn't exist in the device.
- If a method uses server groups, then those server group references should be available. Also, that server group must be present in the database.

Parameters

- opContext—Operational context
- arRuleCol—List of modified AuthorizationRule objects that will replace the existing objects.

Return Value

- void

modifyGlobalAaaServerSetting

Modifies one or more existing global AAA Server setting in a network element.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameId is null or it is not type InstanceNameId.
- If the setting is null.
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the given server type is TACACS+ and TACACS+ is disabled in the given network element that is running Cisco NX-OS.
modifyGlobalAaaServers

Modifies one or more existing Global AAA Servers in a network element.

ValidationException is thrown if any of the following situation occurs:
- If the aaaServerCol is null.
- If the aaaServerCol contains one or more null element, or the collection is empty
- If AAA is disabled in the given network element that is running Cisco IOS.
- If the given AAA server is TACACS+ server and TACACS+ is disabled in the given network element that is running Cisco NX-OS.

PropertiesException is thrown if any of the following situation occurs:
- If AAA server is not valid.

Example:
- Server host name and Ip Address is updated

IntegrityException is thrown if any of the following situation occurs:
- If the given AAA Server doesn't exist globally in the device.

unbindNetworkInterfaceFromAaaServerGroups

Clears the source interface association from the given AAA server groups. If the AAA server groups passed to this API has any source interface association, then those associations will be removed. If they don't have any interface associations, then this API will simply ignore those server groups.

ValidationException is thrown if any of the following situation occurs:
- If aaaServerGroupInstanceNameIdColis null
unbindNetworkInterfaceFromGlobalRadiusServerSettings

Clears the source interface association from the RADIUS server's global setting of a given network element. If the setting does not have any interface association, then this API will simply ignore that network element.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null
- If AAA is disabled in the given network element that is running Cisco IOS.

IntegrityException is thrown if any of the following situation occurs:

- If the object for the given neInstanceNameId doesn't exist in the Database.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the Network Element.

Return Value

- void

unbindNetworkInterfaceFromGlobalTACACSServerSettings

Clears the source interface association from the TACACS server's global setting of a given network element. If the setting does not have any interface association, then this API will simply ignore that network element.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the Network Element.

Return Value

- void

unbindNetworkInterfaceFromGlobalRadiusServerSettings

Clears the source interface association from the RADIUS server's global setting of a given network element. If the setting does not have any interface association, then this API will simply ignore that network element.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null
- If AAA is disabled in the given network element that is running Cisco IOS.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the Network Element.

Return Value

- void
unbindNetworkInterfaceFromGlobalTACACSServerSettings

- If TACACS is disabled in the given network element that is running Cisco NX-OS. IntegrityException is thrown if any of the following situation occurs:
  - If the object for the given neInstanceId doesn't exist in the Database.

**Parameters**

opContext—Operational context
neInstanceId—InstanceNameId of the Network Element.

**Return Value**

void
AclApp Service

This chapter describes the DCNM web services’ API methods for the AclApp service.

Information About AclApp Service

An access control lists (ACLs) allows you to perform packet classification and filtering. An ACL is an ordered set of access control rules. An access control rule is commonly referred to as an access control entry (ACE). Each ACE specifies a packet matching criteria and an action.

Matching criteria can be defined based on packet parameters such as the source address, and the destination address and protocols. An action permits or denies packets that match the specified criteria. A packet may match multiple ACEs in an ACL, but only the first matching ACE is considered. If a packet does not match any of the ACEs, then the packet is dropped.

An ACL must be attached to a target for the target to be activated. The target can be a Layer 3 interface, a VLAN, or a Layer 2 interface.

The API categories are as follows:
- IPv4 ACL—An IPv4 ACL is used to refer the following:
  - Standard IP access lists—Classify or filter traffic using a source address (IPv4 addresses).
  - Extended IP access lists—Classify or filter traffic using a source and destination address (IPv4 addresses) and optional protocol type information.
- MAC ACL—Classify or filter traffic using source and destination MAC addresses and optional protocol type information.
- IPv6 ACL—Classify or filter IPv6 traffic using source and destination addresses and optional protocol type information.
- VLAN Access Map—Provide access control for all packets that are bridged within a VLAN or that are routed into or out of a VLAN.
- Role-based access control lists (RBACL)—Provide access control within a Cisco TrustSec (CTS) domain.

ACL service APIs are defined for the following categories:
- Query/Get APIs—Query data from the persistent database.
- Create APIs—Create new ACLs.
- Modify APIs—Modify existing ACLs.
- Delete APIs—Delete existing ACLs.
• Bind and Unbind APIs—Bind and unbind an association between ACLs and other features.

addRedirectNetworkInterfacesToVlanAccessMapEntry
Assigns one or more network interfaces as redirect interfaces of a VACE. If there are already some interfaces used as redirect interfaces in that VACE, then these interfaces will be added to the existing list of redirected interfaces.

ValidationException is thrown if any of the following situation occurs:
• If vlanAccessMapInstanceId is null or it is not of type VlanAccessMapEntry InstanceNameId.
• If vlanAccessMapInstanceId is not a valid VlanAccessMapEntry InstanceNameId.
• If the networkInterfaceInstanceIdCol collection is null or the collection is empty.
• If the networkInterfaceInstanceIdCol collection contains any null element, or the collection contains an invalid NetworkInterface InstanceNameId.

Parameters
opContext—Operational context.
vlanAccessMapEntryInstanceId—InstanceNameId of VlanAccessMapEntry object.
networkInterfaceInstanceIdCol—a collection of InstanceNameId of NetworkInterface objects to which the classified/filtered traffic needs to be redirected.

Return Value
void

bindIpv4AclToNetworkInterfaces
Assigns an IPv4 ACL to one or more network interfaces in a specified direction.

ParameterException is thrown if any of the following situation occurs:
• If ipv4AclInstanceId is null or it is empty.
• If ipv4AclInstanceId is not a valid StandardAccessControlList or ExtendedAccessControlList InstanceNameId.
• If the networkInterfaceInstanceIdCol collection is null or the collection is empty.
• If the networkInterfaceInstanceIdCol collection contains any null element, or the collection contains an invalid InstanceNameId of a NetworkInterface.
• If the direction is null.

Parameters
opContext—Operational context.
networkInterfaceInstanceIdCol—a collection that contains one or more InstanceNameId of NetworkInterface object.
ipv4AclInstanceId—InstanceNameId of a StandardAccessControlList or ExtendedAccessControlList object.
direction—direction of the network interface traffic, on which the IPv4 ACL needs to be applied. Direction can be either "IN" or "OUT".

Return Value
A list of newly created AclAppliesToNetworkInterface objects.

**bindIpv4AclsToVlanAccessMapEntry**

Applies one or more IPv4 ACLs to a VACE to filter/classify traffic. If there are already some IPv4 ACLs assigned to that VACE, then these IPv4 ACLs will be added to the existing list.

ValidationException is thrown if any of the following situation occurs:
- If vlanAccessMapInstanceNameId is null or it is not of type VlanAccessMapEntry InstanceNameId.
- If vlanAccessMapInstanceNameId is not a valid VlanAccessMapEntry InstanceNameId.
- If the ipv4AclInstanceNameIdCol collection is null or the collection is empty.
- If the ipv4AclInstanceNameIdCol collection contains any null element, or the collection contains an invalid StandardAccessControlEntry or ExtendedAccessControlList InstanceNameId.

Parameters
- opContext—Operational context.
- vlanAccessMapEntryInstanceNameId—InstanceNameId of VlanAccessMapEntry object.
- ipv4AclInstanceNameIdCol—a collection of InstanceNameId of one or more StandardAccessControlList or ExtendedAccessControlList objects.

Return Value
void

**bindIpv6AclToNetworkInterfaces**

Assigns an IPv6 ACL to one or more network interfaces in a specified direction.

ValidationException is thrown if any of the following situation occurs:
- If ipv6AclInstanceNameId is null or it is not of type Ipv6AccessControlList InstanceNameId.
- If ipv6AclInstanceNameId is not a valid Ipv6AccessControlList InstanceNameId.
- If the networkInterfaceInstanceNameIdCol collection is null or the collection is empty.
- If the networkInterfaceInstanceNameIdCol collection contains any null element, or the collection contains an invalid NetworkInterface InstanceNameId.
- If the direction is null.

Parameters
- opContext—Operational context.
- networkInterfaceInstanceNameIdCol—a collection that contains one or more InstanceNameId of NetworkInterface object.
- ipv6AclInstanceNameId—InstanceNameId of a Ipv6AccessControlList object.
bindIpv6AclsToVlanAccessMapEntry

Applies one or more IPv6 ACLs to a VACE to filter/classify traffic. If there are already some IPv6 ACLs assigned to that VACE, then these IPv6 ACLs will be added to the existing list.

ValidationException is thrown if any of the following situation occurs:

- If `vlanAccessMapInstanceNameId` is null or it is not of type `VlanAccessMapEntry InstanceNameId`.
- If `vlanAccessMapInstanceNameId` is not a valid `VlanAccessMapEntry InstanceNameId`.
- If the `ipv6AclInstanceNameIdCol` collection is null or the collection is empty.
- If the `ipv6AclInstanceNameIdCol` collection contains any null element, or the collection contains an invalid `Ipv6AccessControlList InstanceNameId`.

**Parameters**

- `opContext`—Operational context.
- `vlanAccessMapEntryInstanceNameId`—InstanceNameId of `VlanAccessMapEntry` object.
- `ipv6AclInstanceNameIdCol`—a collection of `InstanceNameId` of one or more `Ipv6AccessControlList` objects.

**Return Value**

void

bindMacAclToNetworkInterfaces

Assigns an MAC ACL to one or more network interfaces in a specified direction.

ValidationException is thrown if any of the following situation occurs:

- If `macAclInstanceNameId` is null or it is not of type `MacAccessControlList InstanceNameId`.
- If `macAclInstanceNameId` is not a valid `MacAccessControlList InstanceNameId`.
- If the `networkInterfaceInstanceNameIdCol` collection is null or the collection is empty.
- If the `networkInterfaceInstanceNameIdCol` collection contains any null element, or the collection contains an invalid `NetworkInterface InstanceNameId`.
- If the direction is null.

**Parameters**

- `opContext`—Operational context.
- `networkInterfaceInstanceNameIdCol`—a collection that contains one or more `InstanceNameId` of `NetworkInterface` object.
- `macAclInstanceNameId`—`InstanceNameId` of a `MacAccessControlList` object.
direction—direction of the network interface traffic, on which the MAC ACL needs to be applied.

Return Value
A list of newly created AclAppliesToNetworkInterface objects.

**bindMacAclsToVlanAccessMapEntry**

Applies one or more MAC ACLs to a VACE to filter/classify traffic. If there are already some MAC ACLs assigned to that VACE, then these MAC ACLs will be added to the existing list.

ValidationException is thrown if any of the following situation occurs:

- If vlanAccessMapInstanceNameId is null or it is not of type VlanAccessMapEntry InstanceNameId.
- If vlanAccessMapInstanceNameId is not a valid VlanAccessMapEntry InstanceNameId.
- If the macAclInstanceNameIdCol collection is null or the collection is empty.
- If the macAclInstanceNameIdCol collection contains any null element, or the collection contains an invalid MacAccessControlList InstanceNameId.

**Parameters**

- opContext—Operational context.
- vlanAccessMapEntryInstanceNameId—InstanceNameId of VlanAccessMapEntry object.
- macAclInstanceNameIdCol—a collection of InstanceNameId of one or more MacAccessControlList objects.

**Return Value**

void

**bindTimeRangeToAces**

Assigns a time range to one or more ACEs.

ValidationException is thrown if any of the following situation occurs:

- If timerangeInstanceNameId is null or it is not of type TimeRange InstanceNameId.
- If timerangeInstanceNameId is not a valid TimeRange InstanceNameId.
- If the aceInstanceNameIdCol collection is null or the collection is empty.
- If the aceInstanceNameIdCol collection contains any null element, or the collection contains an invalid ExtendedAccessControlList, Ipv6AccessControlList or RoleBasedAccessControlList InstanceNameId.
- If the direction is null.

**Parameters**

- opContext—Operational context.
- aceInstanceNameIdCol—List of Extended ACEs, Ipv6 ACEs or Role Based ACEs on which time range need to be applied.
- timerangeInstanceNameId—Instance name ID of a timerange object.
**bindVlanAccessMapToVlans**

Assigns an VACL to one or more VLANs.

ValidationException is thrown if any of the following situation occurs:
- If vlanAccessMapEntryInstanceNameId is null or it is not of type VlanAccessMap InstanceNameId.
- If vlanAccessMapEntryInstanceNameId is not a valid VlanAccessMap InstanceNameId.
- If the vlanIds is null.

**Parameters**
- opContext—Operational context.
- vlanAccessMapInstanceNameId—InstanceNameId of a VlanAccessMap object.
- vlanIds—One or more VLAN ID, that uniquely identifies a VLAN.

**Return Value**
void

---

**createExtendedIpAcls**

Creates one or more Extended IP ACL objects in a network element. Given the InstanceNameId of a network element and a list of Extended IP ACL objects, creates the objects in the server and returns it's instance name IDs.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the extendedIpAclCol is null or the collection is empty.
- If the extendedIpAclCol contains one or more null element, or the collection contains objects that are not of type ExtendedAccessControlList.
- If the ExtendedAccessControlEntry, inside the ExtendedAccessControlList does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:
- In the extendedIpAclCol collection, if any of the ExtendedAccessControlList attribute is not valid or the ExtendedAccessControlEntry inside a Extended ACL is not valid.

**Example:**
- name of an ACL starts with a number. Because, an ACL name, cannot contain a space or quotation mark, and must begin with an alphabetic character to prevent ambiguity with numbered access lists.
- remark attribute value of an ACL contains more than 100 characters.

IntegrityException is thrown if any of the following situation occurs:
- If the extendedIpAclCol contains a ExtendedAccessControlList that already exist in the database.
If a ExtendedAccessControlList in the extendedIpAclCol contains duplicate ExtendedAccessControlEntry objects.

This API will not consider the interface association. If a Extended ACL is passed with the interface association, that will not be considered by this API. User needs to call separate API to bind the Extended ACL to an interface.

**Parameters**
- `opContext`—Operational context.
- `neInstanceNameId`—InstanceNameId of a network element.
- `extendedIpAclCol`—a collection (one or more) of Extended ACL objects that needs to be created.

**Return Value**
Instance name IDs of the newly created Extended ACL objects.

---

**createIpv6Acls**

Creates one or more IPv6 ACL objects in a network element. Given the `InstanceNameId` of a network element and a list of IPv6 ACL objects, creates the objects in the server and returns it's instance name IDs.

ValidationException is thrown if any of the following situation occurs:
- If `neInstanceNameId` is null.
- If `neInstanceNameId` is not a valid network element `InstanceNameId`.
- If the `ipv6AclCol` is null or the collection is empty.
- If the `ipv6AclCol` contains one or more null element, or the collection contains objects that are not of type `Ipv6AccessControlList`.
- If the `Ipv6AccessControlEntry`, inside the `Ipv6AccessControlList` does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:
- In the `ipv6AclCol` collection, if any of the `Ipv6AccessControlList` attribute is not valid or the `Ipv6AccessControlEntry` inside a IPv6 ACL is not valid.

Example:
- name of an ACL starts with a number. Because, an ACL name, cannot contain a space or quotation mark, and must begin with an alphabetic character to prevent ambiguity with numbered access lists.
- remark attribute value of an ACL contains more than 100 characters.

IntegrityException is thrown if any of the following situation occurs:
- If the `ipv6AclCol` contains a `Ipv6AccessControlList` that already exist in the database.
- If a `Ipv6AccessControlList` in the `ipv6AclCol` contains duplicate `Ipv6AccessControlEntry` objects.

This API will not consider the interface association. If a IPv6 ACL is passed with the interface association, that will not be considered by this API. User needs to call separate API to bind the IPv6 ACL to an interface.

**Parameters**
- `opContext`—Operational context.
createMacAcls

Creates one or more MAC ACL objects in a network element. Given the InstanceNameId of a network element and a list of MAC ACL objects, creates the objects in the server and returns it's instance name IDs.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the macAclCol is null or the collection is empty.
- If the macAclCol contains one or more null element, or the collection contains objects that are not of type MacAccessControlList.
- If the MacAccessControlEntry, inside the MacAccessControlList does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:

- If the macAclCol collection, if any of the MacAccessControlList attribute is not valid or the MacAccessControlEntry inside a MAC ACL is not valid.

Example:

- name of an ACL starts with a number. Because, an ACL name, cannot contain a space or quotation mark, and must begin with an alphabetic character to prevent ambiguity with numbered access lists.
- remark attribute value of an ACL contains more than 100 characters.

IntegrityException is thrown if any of the following situation occurs:

- If the macAclCol contains a MacAccessControlList that already exist in the database.
- If a MacAccessControlList in the macAclCol contains duplicate MacAccessControlEntry objects.

This API will not consider the interface association. If a MAC ACL is passed with the interface association, that will not be considered by this API. User needs to call separate API to bind the MAC ACL to an interface.

Parameters

- opContext—Operational context.
- neInstanceNameId—InstanceNameId of a network element.
- macAclCol—a collection (one or more) of MAC ACL objects that needs to be created.

Return Value

Instance name IDs of the newly created MAC ACL objects.
createRbacPolicies

Creates one or more Role Based ACL Policy objects in a network element. Given the InstanceNameId of a network element and a list of Role Based ACL Policy objects, creates the objects in the server and returns it's instance name IDs.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the rbaclPolicyCol is null or the collection is empty.
- If the rbaclPolicyCol contains one or more null element, or the collection contains objects that are not of type RoleBasedAccessControlPolicy.

PropertiesException is thrown if any of the following situation occurs:

- In the rbaclPolicyCol collection, if any of the RoleBasedAccessControlPolicy attribute is not valid or the RoleBasedAccessControlEntry inside a Role Based ACL Policy is not valid.

IntegrityException is thrown if any of the following situation occurs:

- If the rbaclPolicyCol contains a RoleBasedAccessControlPolicy that already exist in the database.
- If a RoleBasedAccessControlPolicy in the rbaclPolicyCol contains duplicate elements.

**Parameters**

- opContext—Operational context.
- neInstanceNameId—InstanceNameId of a network element.
- rbaclPolicyCol—a collection (one or more) of RoleBasedAccessControlPolicy objects that needs to be created.

**Return Value**

Instance name IDs of the newly created RoleBasedAccessControlPolicy objects.

createRbacils

Creates one or more RBACL objects in a network element. Given the InstanceNameId of a network element and a list of RBACL objects, creates the objects in the server and returns it's instance name IDs.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the rbacCol is null or the collection is empty.
- If the rbacCol contains one or more null element, or the collection contains objects that are not of type RoleBasedAccessControlList.
- If the RoleBasedAccessControlEntry, inside the RoleBasedAccessControlList does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:

- In the rbacCol collection, if any of the RoleBasedAccessControlList attribute is not valid or the RoleBasedAccessControlEntry inside a RBACL is not valid.
createStandardIpAcls

Creates one or more standard IP ACL objects in a network element. Given the InstanceNameId of a network element and a list of StandardAccessControlList objects, creates the objects in the server and returns the created StandardAccessControlList objects.

InstanceException is thrown if any of the following situation occurs:

- If the neInstanceId is null.
- If the neInstanceId is not a valid InstanceNameId of a network element.

ParameterException is thrown if any of the following situation occurs:

- If the standardIpAclCol is null or the collection is empty.
- If the standardIpAclCol contains one or more null element, or the collection contains objects that are not of type StandardAccessControlList.
- If the StandardAccessControlEntry, inside the StandardAccessControlList does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:

- In the standardIpAclCol collection, if any of the StandardAccessControlList attribute is not valid or the StandardAccessControlEntry inside a StandardAccessControlList is not valid.

Example:

- name of an ACL starts with a number. Because, an ACL name, cannot contain a space or quotation mark, and must begin with an alphabetic character to prevent ambiguity with numbered access lists.
- Remark description value of an ACL has more than 100 characters.


**createTimeRanges**

Creates one or more TimeRange objects in a network element. Given the InstanceNameId of a network element and a list of TimeRange objects, creates the objects in the server and returns its instance name IDs.

ValidationException is thrown if any of the following situation occurs:
- If `neInstanceNameId` is null.
- If `neInstanceNameId` is not a valid network element `InstanceNameId`.
- If the `timerangeCol` is null or the collection is empty.
- If the `timerangeCol` contains one or more null element, or the collection contains objects that are not of type `TimeRange`.

PropertiesException is thrown if any of the following situation occurs:
- In the `timerangeCol` collection, if any of the `TimeRange` attribute is not valid or the `PeriodicTimeRange` inside a `TimeRange` is not valid.

Example:
- name of a `TimeRange` starts with a number. Because, an `TimeRange` name, cannot contain a space or quotation mark, and must begin with an alphabetic character to prevent ambiguity with numbered access lists.

IntegrityException is thrown if any of the following situation occurs:
- If the `timerangeCol` contains a `TimeRange` that already exist in the database.
- If a `TimeRange` in the `timerangeCol` contains duplicate `PeriodicTimeRange` objects.

This API will not consider the ACE association. If a `TimeRange` is passed with the ACE association, that will not be considered by this API. User needs to call separate API to bind the `TimeRange` to an ACE.

**Parameters**
- `opContext`—Operational context.
- `neInstanceNameId`—`InstanceNameId` of a network element.
- `timerangeCol`—A collection (one or more) of `TimeRange` objects that will be created in the database.

**Return Value**
- `TimeRange` objects that are newly created.
createVlanAccessMaps

Creates one or more VACL objects in a network element. Given the InstanceNameId of a network element and a list of VACL objects, creates the objects in the server and returns its instance name IDs.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the vlanAccessMapEntryCol is null or the collection is empty.
- If the vlanAccessMapEntryCol contains one or more null element, or the collection contains objects that are not of type VlanAccessMap.
- If the VlanAccessMapEntry, inside the VlanAccessMap does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:

- In the vlanAccessMapEntryCol collection, if any of the VlanAccessMap attribute is not valid or the VlanAccessMapEntry inside a VACL is not valid.

Example:

- name of an ACL starts with a number. Because, an ACL name, cannot contain a space or quotation mark, and must begin with an alphabetic character to prevent ambiguity with numbered access lists.
- remark attribute value of an ACL contains more than 100 characters.

IntegrityException is thrown if any of the following situation occurs:

- If the vlanAccessMapEntryCol contains a VlanAccessMap that already exist in the database.
- If a VlanAccessMap in the vlanAccessMapEntryCol contains duplicate VlanAccessMapEntry objects.

This API will not consider the interface association. If a VACL is passed with the interface association, that will not be considered by this API. User needs to call separate API to bind the VACL to an interface.

Parameters

- opContext—Operational context.
- neInstanceNameId—InstanceNameId of a network element.
- vlanAccessMapCol—a collection (one or more) of VACL objects that needs to be created.

Return Value

Instance name IDs of the newly created VACL objects.

deleteExtendedIpAcls

Deletes one or more Extended IP ACL objects. Given the InstanceNameId of the ExtendedAccessControlList objects, those objects will be deleted from the server.
ValidationException is thrown if any of the following situation occurs:

- If `extendedIpAclInstanceNameIdCol` collection is null or it is empty.
- If `extendedIpAclInstanceNameIdCol` collection contains an element that is not of type `ExtendedAccessControlList InstanceNameId`.
- If `extendedIpAclInstanceNameIdCol` collection contains a `ExtendedAccessControlList` that does not exist in the database.

**Parameters**

- `opContext`—Operational context.
- `extendedIpAclInstanceNameIdCol`—a collection that contains `InstanceNameId` of one or more `ExtendedAccessControlList` objects that needs to be deleted.

**Return Value**

`void`

---

**deleteIpv6Acls**

Deletes one or more IPv6 ACL objects. Given the `InstanceNameId` of the `Ipv6AccessControlList` objects, those objects will be deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If `ipv6AclInstanceNameIdCol` collection is null or it is empty.
- If `ipv6AclInstanceNameIdCol` collection contains an element that is not of type `Ipv6AccessControlList InstanceNameId`.
- If `ipv6AclInstanceNameIdCol` collection contains a `Ipv6AccessControlList` that does not exist in the database.

**Parameters**

- `opContext`—Operational context.
- `ipv6AclInstanceNameIdCol`—a collection that contains `InstanceNameId` of one or more `Ipv6AccessControlList` objects that needs to be deleted.

**Return Value**

`void`

---

**deleteMacAcls**

Deletes one or more MAC ACL objects. Given the `InstanceNameId` of the `MacAccessControlList` objects, those objects will be deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If `macAclInstanceNameIdCol` collection is null or it is empty.
- If `macAclInstanceNameIdCol` collection contains an element that is not of type `MacAccessControlList InstanceNameId`.
If macAclInstanceNameIdCol collection contains a MacAccessControlList that does not exist in the database.

Parameters

- opContext—Operational context.
- macAclInstanceNameIdCol—a collection that contains InstanceNameId of one or more MacAccessControlList objects that needs to be deleted.

Return Value

void

### deleteRbacIPolicies

Deletes one or more Role based ACL Policy objects. Given the InstanceNameId of the RoleBasedAccessControlPolicy objects, those objects will be deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If rbaclPolicyInstanceNameIdCol collection is null or it is empty.
- If rbaclPolicyInstanceNameIdCol collection contains an element that is not of type RoleBasedAccessControlPolicy InstanceNameId.
- If rbaclPolicyInstanceNameIdCol collection contains a RoleBasedAccessControlPolicy that does not exist in the database.

Parameters

- opContext—Operational context.
- rbaclPolicyInstanceNameIdCol—a collection that contains InstanceNameId of one or more RoleBasedAccessControlPolicy objects that needs to be deleted.

Return Value

void

### deleteRbacls

Deletes one or more Role based ACL objects. Given the InstanceNameId of the RoleBasedAccessControlList objects, those objects will be deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If rbaclInstanceNameIdCol collection is null or it is empty.
- If rbaclInstanceNameIdCol collection contains an element that is not of type RoleBasedAccessControlList InstanceNameId.
- If rbaclInstanceNameIdCol collection contains a RoleBasedAccessControlList that does not exist in the database.

Parameters

- opContext—Operational context.
**deleteStandardIpAcls**

Deletes one or more standard IP ACL objects. Given the InstanceNameId of the StandardAccessControlList objects, those objects will be deleted from the server.

ParameterException is thrown if any of the following situation occurs:

- If standardIpAclInstanceNameIdCol collection is null or it is empty.
- If standardIpAclInstanceNameIdCol collection contains an element that is not of type StandardAccessControlList InstanceNameId.
- If standardIpAclInstanceNameIdCol collection contains a StandardAccessControlList object that does not exist in the database.

**Parameters**

- opContext—Operational context.
- standardIpAclInstanceNameIdCol—a collection that contains InstanceNameId of one or more StandardAccessControlList objects that needs to be deleted.

**Return Value**

void

---

**deleteTimeRanges**

Deletes one or more TimeRange objects. Given the InstanceNameId of the TimeRange objects, those objects will be deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If timerangeInstanceNameIdCol collection is null or it is empty.
- If timerangeInstanceNameIdCol collection contains an element that is not of type TimeRange InstanceNameId.
- If timerangeInstanceNameIdCol collection contains a TimeRange object that does not exist in the database.

**Parameters**

- opContext—Operational context.
- timerangeInstanceNameIdCol—a collection that contains InstanceNameId of one or more TimeRange objects that needs to be deleted.

**Return Value**

void
deleteVlanAccessMaps

Deletes one or more VAACL objects. Given the InstanceNameId of the VlanAccessMap objects, those objects will be deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If vlanAccessMapEntryInstanceNameIdCol collection is null or it is empty.
- If vlanAccessMapEntryInstanceNameIdCol collection contains an element that is not of type VlanAccessMap InstanceNameId.
- If vlanAccessMapEntryInstanceNameIdCol collection contains a VlanAccessMap that does not exist in the database.

**Parameters**

- opContext—Operational context.
- vlanAccessMapInstanceNameIdCol—a collection that contains InstanceNameId of one or more VlanAccessMap objects that needs to be deleted.

**Return Value**

void

getAssociatedVlanAccessMap

Returns VlanAccessMap associated to a VLAN. Given a network element InstanceNameId and a VLAN ID in the network element, returns Vlan AccessMap object associated to that VLAN.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

**Parameters**

- opContext—Operational context.
- neInstanceNameId—InstanceNameId of a network element.
- vlanId—ID of a VLAN.

**Return Value**

VlanAccessMap object associated to the VLAN with given VLAN ID. In the returned VlanAccessMap object, only the following associations will be present, and all other associations will be cleared.

- All associated VlanAccessMapEntry in the returned VlanAccessMap object.
- IP ACL/MAC ACL/IPv6 ACL objects associated with each VlanAccessMapEntry (as these ACLs are used as match conditions in VlanAccessMapEntry), if any, in VlanAccessMap. In IP/MAC/IPv6 ACLs all associations will be cleared, except that of VlanAccessMapEntry.
- NetworkInterface objects associated with each VlanAccessMapEntry, if any, as redirect interfaces. In the network interfaces all associations will be cleared, except that of VlanAccessMapEntry.

Returns null if no VlanAccessMap is associated to that VLAN.
getExtendedIpAclToNetworkInterfaceAssociationsInNetworkElement

Returns all ExtendedAccessControlList objects to NetworkInterface associations configured in a network element. Given the InstanceNameId of the network element, returns a collection of ExtendedAccessControlList to NetworkInterface association objects.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
- opContext—Operational context.
- neInstanceNameId—InstanceNameId of the network element.

Return Value
List of AclAppliesToNetworkInterface objects, that represents the association between ExtendedAccessControlList and NetworkInterface. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- ExtendedAccessControlList object. In the ACL object all the associations will be cleared, except AclAppliesToNetworkInterface association.
- NetworkInterface object. In the NetworkInterface object all the associations will be cleared, except AclAppliesToNetworkInterface association.

getExtendedIpAcls

Returns ExtendedAccessControlList objects from it's InstanceNameIds. Given a collection of InstanceNameId of ExtendedAccessControlList, returns corresponding ExtendedAccessControlList objects.

ValidationException is thrown if any of the following situation occurs:
- If extendedIpAclInstanceNameIdCol is null or it is empty.
- If extendedIpAclInstanceNameIdCol contains invalid InstanceNameId of a ExtendedAccessControlList.
- If extendedIpAclInstanceNameIdCol contains a null value.
- If there is no equivalent ExtendedAccessControlList object with the given InstanceNameId in the extendedIpAclInstanceNameIdCol.

Parameters
- opContext—Operational context.
- extendedIpAclInstanceNameIdCol—a collection of InstanceNameId of ExtendedAccessControlList.

Return Value
List of ExtendedAccessControlList objects corresponding to given collection of InstanceNameId. In the returned list of ExtendedAccessControlList objects, only the following associations will be present, and all other associations will be cleared.
- All associated ACEs of the returned ACL object.
getExtendedIpAclsInNetworkElement

Returns all ExtendedAccessControlList objects in a network element. Given the InstanceNameId of a network element, returns a collection of ExtendedAccessControlList objects in the network element. ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of ExtendedAccessControlList objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- All associated ACEs of the returned ACL object.
- TimeRange association for every ACEs, if any, in ExtendedAccessControlList. If that TimeRange has other associations like PeriodicTimeRange entries and so on, those associations will be cleared.

getExtendedNamedIpAclsInNetworkElement

Returns all named ExtendedAccessControlList objects in a network element. Given the InstanceNameId of a network element, returns a collection of ExtendedAccessControlList objects in the network element, that are uniquely identified by it's name.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of named ExtendedAccessControlList objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- All associated ACEs of the returned ACL object.
- TimeRange association for every ACEs, if any, in ExtendedAccessControlList. If that TimeRange has other associations like PeriodicTimeRange entries and so on, those associations will be cleared.
getExtendedNumberedIpAclsInNetworkElement

Returns all numbered ExtendedAccessControlList objects in a network element. Given the InstanceNameId of a network element, returns a collection of ExtendedAccessControlList objects in the network element, that are uniquely identified by it's number.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of numbered ExtendedAccessControlList objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- All associated ACEs of the returned ACL object.
- TimeRange association for every ACEs, if any, in ExtendedAccessControlList. If that TimeRange has other associations like PeriodicTimeRange entries and so on, those associations will be cleared.

getIpAclToNetworkInterfaceAssociationsInNetworkElement

Returns all IP ACLs (IPv4 ACLs and IPv6 ACLs) to network interface associations configured in a network element. Given the instance name ID of the network element, returns a collection of IP ACL to network interface association objects.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of AclAppliesToNetworkInterface objects, that represents the association between Standard/Extended/IPv6 ACL and NetworkInterface. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- StandardAccessControlList, ExtendedAccessControlList or Ipv6AccessControlList object. In the ACL object all the associations will be cleared, except AclAppliesToNetworkInterface association.
- NetworkInterface object. In the NetworkInterface object all the associations will be cleared, except AclAppliesToNetworkInterface association.

getIpAclsInNetworkElement

Returns all IP ACLs (IPv4 ACLs and IPv6 ACLs) configured in a network element. Given the InstanceNameId of the network element, returns a collection of IP ACL objects.
getIpv4AclToNetworkInterfaceAssociationsInNetworkElement

Returns all IPv4 ACLs both StandardAccessControlList and ExtendedAccessControlList objects) to NetworkInterface associations configured in a network element. Given the InstanceNameId of the network element, returns a collection of IPv4 ACL to NetworkInterface association objects.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters

- opContext—Operational context.
- neInstanceNameId—InstanceNameId of the network element.

Return Value

List of AclAppliesToNetworkInterface objects, that represents the association between Standard/Extended ACL and NetworkInterface. NetworkInterface. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- StandardAccessControlList or ExtendedAccessControlList object. In the ACL object all the associations will be cleared, except AclAppliesToNetworkInterface association.
- NetworkInterface object. In the NetworkInterface object all the associations will be cleared, except AclAppliesToNetworkInterface association.

getIpv4Acls

Returns IPv4 ACLs (both StandardAccessControlList and ExtendedAccessControlList objects) from it's InstanceNameIds. Given a collection of InstanceNameId of StandardAccessControlList and ExtendedAccessControlList, returns corresponding StandardAccessControlList and ExtendedAccessControlList objects.
ValidationException is thrown if any of the following situation occurs:

- If ipv4AcInstanceNameIdCol is null or it is empty.
- If ipv4AcInstanceNameIdCol contains invalid InstanceNameId of a StandardAccessControlList or ExtendedAccessControlList.
  - null value.
- If there is no equivalent IPv4 ACL object with the given InstanceNameId in the ipv4AcInstanceNameIdCol.

**Parameters**

- **opContext**—Operational context.
- **ipv4AcInstanceNameIdCol**—a collection of InstanceNameId of StandardAccessControlList and ExtendedAccessControlList.

**Return Value**

List of StandardAccessControlList and ExtendedAccessControlList objects corresponding to given collection of InstanceNameId. In the returned list of IPv4 ACL objects, only the following associations will be present, and all other associations will be cleared.

- All associated ACEs of the returned ACL object.
- TimeRange association for every ACEs, if any, in ExtendedAccessControlList. If that TimeRange has other associations like PeriodicTimeRange entries and so on, those associations will be cleared.

### getIpv4AclsInNetworkElement

Returns all IPv4 ACLs (Standard ACLs and Extended ACLs) configured in a network element. Given the InstanceNameId of the network element, returns a collection of IP ACL objects.

InstanceException is thrown if the argument passed neInstanceNameId is null or it is not a valid network element InstanceNameId.

**Parameters**

- **opContext**—Operational context. operational context.
- **neInstanceNameId**—InstanceNameId of the network element.

**Return Value**

List of StandardAccessControlList, ExtendedAccessControlList and Ipv6AccessControlList objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- All associated ACEs of the returned ACL object.
- All associated remarks of the returned ACL object.
- TimeRange association for every ACEs, if any, in ExtendedAccessControlList and Ipv6AccessControlList. If that TimeRange has other associations like PeriodicTimeRange entries and so on, those associations will be cleared.

Returned ACLs will be ordered by the name or number of the ACL.
getIpv4AclsWithoutAcesInNetworkElement

Returns all IPv4 ACLs (Standard ACLs and Extended ACLs) configured in a network element. Given the InstanceNameId of the network element, returns a collection of IP ACL objects.

InstanceException is thrown if the argument passed neInstanceNameId is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of StandardAccessControlList, ExtendedAccessControlList and Ipv6AccessControlList objects. In the returned list of objects, all associations will be cleared. Returned ACLs will be ordered by the name or number of the ACL.

getIpv6AclToNetworkInterfaceAssociationsInNetworkElement

Returns all Ipv6AccessControlList objects to NetworkInterface associations configured in a network element. Given the InstanceNameId of the network element, returns a collection of Ipv6AccessControlList to NetworkInterface association objects.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of AclAppliesToNetworkInterface objects, that represents the association between Ipv6AccessControlList and NetworkInterface. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- Ipv6AccessControlList object. In the ACL object all the associations will be cleared, except AclAppliesToNetworkInterface association.
- NetworkInterface object. In the NetworkInterface object all the associations will be cleared, except AclAppliesToNetworkInterface association.

getIpv6Acls

Returns Ipv6AccessControlList objects from it's InstanceNameIds. Given a collection of InstanceNameId of Ipv6AccessControlList}, returns corresponding Ipv6AccessControlList objects.

ValidationException is thrown if any of the following situation occurs:

- If ipv6AclInstanceNameIdCol is null or it is empty.
- If ipv6AclInstanceNameIdCol contains invalid InstanceNameId of a Ipv6AccessControlList.
getIpv6AclsInNetworkElement

Returns all IPv6 ACLs in a network element. Given the InstanceNameId of a network element, returns a collection of IPv6 ACLs in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context.
- neInstanceNameId—InstanceNameId of the network element.

Return Value
List of Ipv6AccessControlList objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- All associated ACEs of the returned ACL object.
- TimeRange association for every ACEs, if any, in Ipv6AccessControlList. If that TimeRange has other associations like PeriodicTimeRange entries and so on, those associations will be cleared.

getIpv6AclsWithoutAcesInNetworkElement

Returns all IPv6 ACLs in a network element. Given the InstanceNameId of a network element, returns a collection of IPv6 ACLs in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context.
- neInstanceNameId—InstanceNameId of the network element.

Return Value
List of Ipv6AccessControlList objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- All associated ACEs of the returned ACL object.
- TimeRange association for every ACEs, if any, in Ipv6AccessControlList. If that TimeRange has other associations like PeriodicTimeRange entries and so on, those associations will be cleared.
### Return Value

List of Ipv6AccessControlList objects. In the returned list of objects, all associations will be cleared.

### getMacAclToNetworkInterfaceAssociationsInNetwork Element

Returns all MAC ACLs to network interface associations configured in a network element. Given the instance name ID of the network element, returns a collection of MAC ACL to network interface association objects.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

**Parameters**

- **opContext**—Operational context.
- **neInstanceNameId**—InstanceNameId of the network element.

**Return Value**

List of AclAppliesToNetworkInterface objects, that represents the association between MAC ACL and network interface. In the returned list of objects, only the following associations will be present, and all other associations will be cleared:

- MacAccessControlList object. In the ACL object all the associations will be cleared, except AclAppliesToNetworkInterface association.
- NetworkInterface object. In the network interface object all the associations will be cleared, except AclAppliesToNetworkInterface association.

### getMacAcls

Returns MacAccessControlList objects from it's InstanceNameIds. Given a collection of InstanceNameId of MacAccessControlList, returns corresponding MacAccessControlList objects.

ValidationException is thrown if any of the following situation occurs:

- If macAclInstanceNameIdCol is null or it is empty.
- If macAclInstanceNameIdCol contains invalid InstanceNameId of a MacAccessControlList.
- If macAclInstanceNameIdCol contains null values.
- If there is no equivalent MacAccessControlList object with the given InstanceNameId in the macAclInstanceNameIdCol.

**Parameters**

- **opContext**—Operational context.
- **macAclInstanceNameIdCol**—a collection of InstanceNameId of MacAccessControlList.
Return Value
List of MacAccessControlList objects corresponding to given collection of InstanceNameId. In the returned list of MacAccessControlList objects, only the MacAccessControlEntries associated with returned MacAccessControlList objects will be present, and all other associations will be cleared.

getMacAclsInNetworkElement

Returns all MacAccessControlList objects in a network element. Given the InstanceNameId of a network element, returns a collection of MacAccessControlList in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of MacAccessControlList objects. In the returned list of MacAccessControlList objects, only the MAC ACEs associated with returned MacAccessControlList objects will be present, and all other associations will be cleared.

getMacAclsWithoutAcesInNetworkElement

Returns all MacAccessControlList objects in a network element. Given the InstanceNameId of a network element, returns a collection of MacAccessControlList in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of MacAccessControlList objects. In the returned list of MacAccessControlList objects, all associations will be cleared.

getNamedIpv4AclsInNetworkElement

Returns all named IPv4 ACLs (both StandardAccessControlList and ExtendedAccessControlList objects) in a network element. Given the InstanceNameId of a network element, returns a collection of IPv4 ACLs in the network element, that are uniquely identified by it’s name.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.
getNumberedIpv4AclsInNetworkElement

Returns all numbered IPv4 ACLs (both StandardAccessControlList and ExtendedAccessControlList) objects in a network element. Given the InstanceNameId of a network element, returns a collection of IPv4 ACLs in the network element, that are uniquely identified by it's number.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters

opContext—Operational context.

neInstanceNameId—InstanceNameId of the network element.

Return Value

List of numbered StandardAccessControlList and ExtendedAccessControlList objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- All associated ACEs of the returned ACL object.
- TimeRange association for every ACEs, if any, in ExtendedAccessControlList. If that TimeRange has other associations like PeriodicTimeRange entries and so on, those associations will be cleared.

getRbacPolicies

Returns RBACL policies from it's InstanceNameIds. Given a collection of InstanceNameId of RoleBasedAccessControlPolicy, returns corresponding RBACL policies objects.

ValidationException is thrown if any of the following situation occurs:

- If rbaclPolicyInstanceNameIdCol is null or it is empty.
- If rbaclPolicyInstanceNameIdCol contains invalid Role Based ACL InstanceNameId or null value.
- If there is no equivalent RBACL Policy object with the given InstanceNameId in the rbaclPolicyInstanceNameIdCol.

Parameters

opContext—Operational context.

rbaclPolicyInstanceNameIdCol—a collection of InstanceNameId of RoleBasedAccessControlPolicy.
getRbaclPoliciesInNetworkElement

Returns all Role Based ACL policies in a network element. Given the InstanceNameId of a network element, returns a collection of Role Based ACL policies in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of RoleBasedAccessControlPolicy objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- All associated RoleBasedAccessControlList objects. In the RBACL objects all the associations will be cleared, except that of RBACL Policies.

getRbacls

Returns Role Based ACLs from it's InstanceNameIds. Given a collection of InstanceNameId of RoleBasedAccessControlList, returns corresponding Role Based ACL objects.

ValidationException is thrown if any of the following situation occurs:

- If rbacIInstanceNameIdCol is null or it is empty.
- If rbacIInstanceNameIdCol contains invalid Role Based ACL InstanceNameId or null value.
- If there is no equivalent Role Based ACL object with the given InstanceNameId in the rbacIInstanceNameIdCol.

Parameters
opContext—Operational context.
rbacIInstanceNameIdCol—a collection of InstanceNameId of RoleBasedAccessControlList.

Return Value
List of RoleBasedAccessControlList objects corresponding to given collection of InstanceNameId. In the returned list of Role Based ACL objects, only the following associations will be present, and all other associations will be cleared.

- All associated ACEs of the returned ACL object.
getRbaclsInNetworkElement

Returns all Role Based ACLs in a network element. Given the InstanceNameId of a network element, returns a collection of Role Based ACLs in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

- opContext—Operational context.
- neInstanceNameId—InstanceNameId of the network element.

Return Value

List of RoleBasedAccessControlList objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- All associated ACEs of the returned ACL object.
- Timerange association for every ACEs, if any, in RoleBasedAccessControlList. If that TimeRange has other associations like periodic TimeRange entries and so on, those associations will be cleared.

getStandardIpAclToNetworkInterfaceAssociationsInNetworkElement

Returns all StandardAccessControlList objects to NetworkInterface associations configured in a network element. Given the InstanceNameId of the network element, returns a collection of StandardAccessControlList to NetworkInterface association objects.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters

- opContext—Operational context.
- neInstanceNameId—InstanceNameId of the network element.

Return Value

List of AclAppliesToNetworkInterface objects, that represents the association between StandardAccessControlList and NetworkInterface. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- StandardAccessControlList object. In the ACL object all the associations will be cleared, except AclAppliesToNetworkInterface association.
- NetworkInterface object. In the network interface object all the associations will be cleared, except AclAppliesToNetworkInterface association.
getStandardIpAcls

Returns StandardAccessControlList objects from it’s InstanceNameIds. Given a collection of InstanceNameId of StandardAccessControlList, returns corresponding StandardAccessControlList objects.

ValidationException is thrown if any of the following situation occurs:
- If standardIpAclInstanceNameIdCol is null or it is empty.
- If standardIpAclInstanceNameIdCol contains invalid InstanceNameId of a StandardAccessControlList.
- If standardIpAclInstanceNameIdCol collection contains a null value.
- If there is no equivalent StandardAccessControlList object with the given InstanceNameId in the standardIpAclInstanceNameIdCol.

Parameters
opContext—Operational context.
standardIpAclInstanceNameIdCol—a collection of InstanceNameId of StandardAccessControlList.

Return Value
List of StandardAccessControlList objects corresponding to given collection of InstanceNameId. In the returned list of StandardAccessControlList objects, only the StandardAccessControlEntry objects associated with returned StandardAccessControlList objects will be present, and all other associations will be cleared.

getStandardIpAclsInNetworkElement

Returns all StandardAccessControlList objects in a network element. Given the InstanceNameId of a network element, returns a collection of StandardAccessControlList objects in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of StandardAccessControlList objects. In the returned list of StandardAccessControlList objects, only the Standard ACEs associated with returned StandardAccessControlList objects will be present, and all other associations will be cleared.

getStandardNamedIpAclsInNetworkElement

Returns all named StandardAccessControlList objects in a network element. Given the InstanceNameId of a network element, returns a collection of StandardAccessControlList objects in the network element, that are uniquely identified by it's name.
getStandardNumberedIpAclsInNetworkElement

Returns all numbered StandardAccessControlList objects in a network element. Given the InstanceNameId of a network element, returns a collection of StandardAccessControlList objects in the network element, that are uniquely identified by its number.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters
opContext—Operational context.
neInstanceId—InstanceNameId of the network element.

Return Value
List of numbered StandardAccessControlList objects. In the returned list of numbered StandardAccessControlList objects, only the StandardAccessControlEntry objects associated with returned StandardAccessControlList objects will be present, and all other associations will be cleared.

getTimeRanges

Returns TimeRanges from its InstanceNameIds. Given a collection of InstanceNameId of TimeRange, returns corresponding TimeRange objects.

ValidationException is thrown if any of the following situation occurs:

- If timerangeInstanceIdCol is null or it is empty.
- If timerangeInstanceIdCol contains invalid Extended ACL InstanceNameId or null value.
- If there is no equivalent TimeRange object with the given InstanceNameId in the timerangeInstanceIdCol.

Parameters
opContext—Operational context.
timerangeInstanceIdCol—a collection of InstanceNameId of TimeRange.
Chapter 3  AclApp Service

getTimeRangesInNetworkElement

Returns all TimeRanges in a network element. Given the InstanceNameId of a network element, returns a collection of TimeRanges in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

- opContext—Operational context.
- neInstanceNameId—InstanceNameId of the network element.

Return Value

List of TimeRange objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- All associated periodic TimeRange entries.
- Absolute TimeRange entry.

getTimeRangesWithoutEntriesInNetworkElement

Returns all TimeRanges in a network element. Given the InstanceNameId of a network element, returns a collection of TimeRanges in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

- opContext—Operational context.
- neInstanceNameId—InstanceNameId of the network element.

Return Value

List of TimeRange objects. In the returned list of objects, all associations will be cleared.

getVlanAccessLogSettingInNetworkElements

Returns VACL log settings applied on the network elements. Given a collection of InstanceNameId of a network element, returns the VACL log settings applied on those network elements.

ValidationException is thrown if any of the following situation occurs:
getVlanAccessMaps

If neInstanceNameIdCol is null or it is empty.
If neInstanceNameIdCol contains invalid network element InstanceNameId or null value.
If there is no equivalent network element object with the given InstanceNameId in the neInstanceNameIdCol.

Parameters
opContext—Operational context.
nInstanceNameIdCol—a collection of InstanceNameId of the network element.

Return Value
List of VlanAccessLog objects corresponding to the InstanceNameId of the network element.

getVlanAccessMaps

Returns VACLs from the InstanceNameIds. Given a collection of InstanceNameId of VlanAccessMap, returns corresponding VACL objects.
ValidationException is thrown if any of the following situation occurs:
- If vlanAccessMapEntryInstanceNameIdCol is null or it is empty.
- If vlanAccessMapEntryInstanceNameIdCol contains invalid VACL InstanceNameId or null value.
- If there is no equivalent VACL object with the given InstanceNameId in the vlanAccessMapEntryInstanceNameIdCol.

Parameters
opContext—Operational context.
vlanAccessMapInstanceNameIdCol—a collection of InstanceNameId of VlanAccessMap.

Return Value
List of VlanAccessMap objects corresponding to given collection of InstanceNameId. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- All associated VACEs in the returned VACL object.
- IP ACL/MAC ACL/IPv6 ACL objects associated with each VACE (as these ACLs are used as match conditions in VACE), if any, in VlanAccessMap. In IP/MAC/IPv6 ACLs all associations will be cleared, except that of VACE.
- NetworkInterface objects associated with each VACE, if any, as redirect interfaces. In the network interfaces all associations will be cleared, except that of VACE.

getVlanAccessMapsInNetworkElement

Returns all VLAN ACLs in a network element. Given the InstanceNameId of a network element, returns a collection of VLAN ACLs in the network element.
ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.
Parameters

opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value

List of VlanAccessMap objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- All associated VACEs in the returned VACL object.
- IP ACL/MAC ACL/IPv6 ACL objects associated with each VACE (as these ACLs are used as match conditions in VACE), if any, in VlanAccessMap. In IP/MAC/IPv6 ACLs all associations will be cleared, except that of VACE.
- NetworkInterface objects associated with each VACE, if any, as redirect interfaces. In the network interfaces all associations will be cleared, except that of VACE.

getVlanAccessMapsWithoutVlanAccessMapEntriesInNetworkElement

Returns all VLAN ACLs in a network element. Given the InstanceNameId of a network element, returns a collection of VLAN ACLs in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

opContext—Operational context.
neInstanceNameId—InstanceNameId of the network element.

Return Value

List of VlanAccessMap objects. In the returned list of objects, all associations will be cleared.

modifyAclSequence

Modifies the sequence number of the ACEs in an ACL, based on the starting sequence number and the step to increment the sequence numbers.

Parameters

opContext—Operational context.
acIInstanceNameIdCol—InstanceNameId of one are more ACLs.

If the platform type is Nexus 7000 series switch, then the ACLs can be of any of the following types:

- StandardAccessControlList
- ExtendedAccessControlList
- MacAccessControlList
- Ipv6AccessControlList
modifyExtendedIpAcls

Modifies one or more existing Extended IP ACL objects.

ValidationException is thrown if any of the following situation occurs:

- If `extendedIpAclCol` collection is null or it is empty.
- If `extendedIpAclCol` collection contains an object that is not of type `ExtendedAccessControlList`.
- If any of the `ExtendedAccessControlEntry`, in the `ExtendedAccessControlList`, does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:

- In the `extendedIpAclCol` collection, if any attribute in the `ExtendedAccessControlList` is not valid or if any `ExtendedAccessControlEntry` inside a Extended ACL is not valid.

Example:

- `seqNo` of an ACE is out of range.
- `remark` attribute value of an ACL contains more than 100 characters.

IntegrityException is thrown if any of the following situation occurs:

- If the `extendedIpAclCol` collection contains a `ExtendedAccessControlList` that does not exist in the database.
- If a `ExtendedAccessControlList` in the `extendedIpAclCol` contains duplicate `ExtendedAccessControlEntry` objects.

This API will not consider the interface association. If a Extended ACL is passed with the interface association, that will not be considered by this API. User needs to call separate API to bind the Extended ACL to an interface.

**Parameters**

- `opContext`—Operational context.
- `extendedIpAclCol`—A collection (one or more) of `ExtendedAccessControlList` objects that will replace the existing `ExtendedAccessControlList` objects in the database.

**Return Value**

`void`
**modifyIpv6Acls**

Modifies one or more existing IPv6 ACL objects.

ValidationException is thrown if any of the following situation occurs:

- If ipv6AclCol collection is null or it is empty.
- If ipv6AclCol collection contains an object that is not of type Ipv6AccessControlList.
- If any of the Ipv6AccessControlEntry, in the Ipv6AccessControlList, does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:

- In the ipv6AclCol collection, if any attribute in the Ipv6AccessControlList is not valid or if any Ipv6AccessControlEntry inside a IPv6 ACL is not valid.

Example:

- seqNo of an ACE is out of range.
- remark attribute value of an ACL contains more than 100 characters.

IntegrityException is thrown if any of the following situation occurs:

- If the ipv6AclCol collection contains a Ipv6AccessControlList that does not exist in the database.
- If a Ipv6AccessControlList in the ipv6AclCol contains duplicate Ipv6AccessControlEntry objects.

This API will not consider the interface association. If a IPv6 ACL is passed with the interface association, that will not be considered by this API. User needs to call separate API to bind the IPv6 ACL to an interface.

**Parameters**

- **opContext**—Operational context.
- **ipv6AclCol**—a collection (one or more) of Ipv6AccessControlList objects that will replace the existing Ipv6AccessControlList objects in the database.

**Return Value**

void

---

**modifyMacAcls**

Modifies one or more existing MAC ACL objects.

ValidationException is thrown if any of the following situation occurs:

- If macAclCol collection is null or it is empty.
- If macAclCol collection contains an object that is not of type MacAccessControlList.
- If any of the MacAccessControlEntry, in the MacAccessControlList, does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:

- In the macAclCol collection, if any attribute in the MacAccessControlList is not valid or if any MacAccessControlEntry inside a MAC ACL is not valid.

Example:
modifyRbaclPolicies

Modifies one or more existing RBACL Policy objects.

ValidationException is thrown if any of the following situation occurs:
- If rbaclPoliyCol collection is null or it is empty.
- If rbaclPoliyCol collection contains an object that is not of type RoleBasedAccessControlPolicy.

PropertiesException is thrown if any of the following situation occurs:
- In the rbaclPoliyCol collection, if any attribute in the RoleBasedAccessControlPolicy is not valid.

Example:
- srcTagType is not specified in RoleBasedAccessControlPolicy.
- sgt is not specified in RoleBasedAccessControlPolicy.

IntegrityException is thrown if any of the following situation occurs:
- If the rbaclPoliyCol collection contains a RoleBasedAccessControlPolicy that does not exist in the database.
- If the rbaclPoliyCol collection contains duplicate RoleBasedAccessControlPolicy objects.

Parameters
opContext—Operational context.
rbaclPoliyCol—a collection (one or more) of RoleBasedAccessControlPolicy objects that will replace the existing RoleBasedAccessControlPolicy objects in the database.

Return Value
void
modifyRbacls

Modifies one or more existing Role Based ACL objects.

ValidationException is thrown if any of the following situation occurs:

- If rbaclCol collection is null or it is empty.
- If rbaclCol collection contains an object that is not of type RoleBasedAccessControlList.
- If any of the RoleBasedAccessControlEntry, in the RoleBasedAccessControlList, does not contain a sequence number.

PropertiesException is thrown if any of the following situation occurs:

- In the rbaclCol collection, if any attribute in the RoleBasedAccessControlList is not valid or if any RoleBasedAccessControlEntry inside a Role Based ACL is not valid.

Example:

- seqNo of an ACE is out of range.
- remark attribute value of an ACL contains more than 100 characters.

IntegrityException is thrown if any of the following situation occurs:

- If the rbaclCol collection contains a RoleBasedAccessControlList that does not exist in the database.
- If a RoleBasedAccessControlList in the rbaclCol contains duplicate RoleBasedAccessControlEntry objects.

Parameters

opContext—Operational context.

rbaclCol—a collection (one or more) of RoleBasedAccessControlList objects that will replace the existing RoleBasedAccessControlList objects in the database.

Return Value

void

modifyStandardIpAcls

Modifies one or more existing StandardAccessControlList objects.

ParameterException is thrown if any of the following situation occurs:

- If standardIpAclCol collection is null or it is empty.
- If standardIpAclCol collection contains an object that is not of type StandardAccessControlList.
- If any of the StandardAccessControlEntry, in the StandardAccessControlList, does not contain sequence number.
- If the standardIpAclCol collection contains a StandardAccessControlList that does not exist in the database.

PropertiesException is thrown if any of the following situation occurs:

- In the standardIpAclCol collection, if any attribute in the StandardAccessControlList is not valid or if any StandardAccessControlEntry inside a Standard ACL is not valid.

Example:
modifyTimeRanges

Modifies one or more existing TimeRange objects.

ValidationException is thrown if any of the following situation occurs:
- If timerangeCol collection is null or it is empty.
- If timerangeCol collection contains an object that is not of type TimeRange.

PropertiesException is thrown if any of the following situation occurs:
- In the timerangeCol collection, if any attribute in the TimeRange is not valid or if any
  PeriodicTimeRange inside a TimeRange is not valid or if any AbsoluteTimeRange inside a
  TimeRange is not valid.

Example:
- Both startTime and endTime is not set in AbsoluteTimeRange.
- endTime is not greater than startTime in AbsoluteTimeRange or PeriodicTimeRange.

IntegrityException is thrown if any of the following situation occurs:
- If the timerangeCol collection contains a TimeRange that does not exist in the database.
- If a TimeRange in the timerangeCol contains duplicate PeriodicTimeRange objects.

This API will not consider the ACE association. If a TimeRange is passed with ACE association, that
will not be considered by this API. User needs to call separate API to bind the TimeRange to an ACE.

Parameters
opContext—Operational context.
timerangeCol—a collection (one or more) of TimeRange objects that will replace the existing
TimeRange objects in the database.

Return Value
void
modifyVlanAccessLogSetting

Modifies VACL log object in a network element.
ValidationException is thrown if any of the following情况 occurs:
- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the vlanAccessMapEntryLog is null.
PropertiesException is thrown if any of the following situation occurs:
- In the vlanAccessMapEntryLog, if any attribute is invalid.
Example:
- maxFlow of a VlanAccessLog is out of range.
- rateLimit of a VlanAccessLog is out of range.

Parameters
- opContext—Operational context.
- neInstanceNameId—InstanceNameId of a network element.
- vlanAccessLog—VlanAccessLog object that will replace the existing VlanAccessLog object in the database.

Return Value
void

modifyVlanAccessMaps

Modifies one or more existing VACL objects.
ValidationException is thrown if any of the following situation occurs:
- If vlanAccessMapEntryCol collection is null or it is empty.
- If vlanAccessMapEntryCol collection contains an object that is not of type VlanAccessMap.
- If any of the VlanAccessMapEntry, in the VlanAccessMap, does not contain a sequence number.
PropertiesException is thrown if any of the following situation occurs:
- In the vlanAccessMapEntryCol collection, if any attribute in the VlanAccessMap is not valid or if any VlanAccessMapEntry inside a VACL is not valid.
Example:
- seqNo of a VlanAccessMapEntry is out of range.
- No IP ACL or MAC ACL or IPv6 ACL is specified in the match condition of any of the VlanAccessMapEntry.
IntegrityException is thrown if any of the following situation occurs:
- If the vlanAccessMapEntryCol collection contains a VlanAccessMap that does not exist in the database.
- If a VlanAccessMap in the vlanAccessMapEntryCol contains duplicate VlanAccessMapEntry objects.
unbindIpv4AclFromNetworkInterface

Clears the IPv4 ACL association from a network interface on a specific direction. ParameterException is thrown if any of the following situation occurs:

- If networkInterfaceInstanceNameId is null or it is not a valid InstanceNameId of a NetworkInterface object.
- If the direction is null.

Parameters

- opContext—Operational context.
- networkInterfaceInstanceNameId—InstanceNameId of a NetworkInterface object from which ACL needs to be removed.
- direction—direction from which StandardAccessControlList or ExtendedAccessControlList object needs to be removed.

Return Value

void

unbindIpv4AclsFromVlanAccessMapEntry

Clears the given list of IPv4 ACLs from the VACE. ValidationException is thrown if any of the following situation occurs:

- If vlanAccessMapInstanceNameId is null or it is not of type VlanAccessMapEntry InstanceNameId.
- If vlanAccessMapInstanceNameId is not a valid VlanAccessMapEntry InstanceNameId.
- If the ipv4AclInstanceNameIdCol collection is null or the collection is empty.
- If the ipv4AclInstanceNameIdCol collection contains any null element, or the collection contains an invalid StandardAccessControlList or ExtendedAccessControlList InstanceNameId.

Parameters

- opContext—Operational context.
- vlanAccessMapEntryInstanceNameId—InstanceNameId of VlanAccessMapEntry object.
- ipv4AclInstanceNameIdCol—a collection of InstanceNameId of one or more StandardAccessControlList or ExtendedAccessControlList objects.
unbindIpv6AclFromNetworkInterface

Clears the IPv6 ACL association from a network interface on a specific direction.

ValidationException is thrown if any of the following situation occurs:

- If networkInterfaceInstanceNameId is null or it is not a valid NetworkInterface object InstanceNameId.
- If the direction is null.

Parameters

- opContext—Operational context.
- networkInterfaceInstanceNameId—InstanceNameId of a NetworkInterface object from which ACL needs to be removed.
- direction—direction from which Ipv6AccessControlList object needs to be removed.

Return Value

void

unbindIpv6AclsFromVlanAccessMapEntry

Clears the given list of IPv6 ACLs from the VACE.

ValidationException is thrown if any of the following situation occurs:

- If vlanAccessMapInstanceNameId is null or it is not of type VlanAccessMapEntry InstanceNameId.
- If vlanAccessMapInstanceNameId is not a valid VlanAccessMapEntry InstanceNameId.
- If the ipv6AclInstanceNameIdCol collection is null or the collection is empty.
- If the ipv6AclInstanceNameIdCol collection contains any null element, or the collection contains an invalid Ipv6AccessControlList InstanceNameId.

Parameters

- opContext—Operational context.
- vlanAccessMapEntryInstanceNameId—InstanceNameId of VlanAccessMapEntry object.
- ipv6AclInstanceNameIdCol—a collection of InstanceNameId of one or more Ipv6AccessControlList objects.

Return Value

void

unbindMacAclFromNetworkInterface

Clears the MAC ACL association from a network interface on a specific direction.
ValidationException is thrown if any of the following situation occurs:

- If networkInterfaceInstanceNameId is null or it is not a valid NetworkInterface object InstanceNameId.
- If the direction is null.

Parameters

opContext—Operational context.

networkInterfaceInstanceNameId—InstanceNameId of a NetworkInterface object from which ACL needs to be removed.

direction—direction from which MacAccessControlList object needs to be removed.

Return Value

void

unbindMacAclsFromVlanAccessMapEntry

Clears the given list of MAC ACLs from the VACE.

ValidationException is thrown if any of the following situation occurs:

- If vlanAccessMapInstanceNameId is null or it is not of type VlanAccessMapEntry InstanceNameId.
- If vlanAccessMapInstanceNameId is not a valid VlanAccessMapEntry InstanceNameId.
- If the macAclInstanceNameIdCol collection is null or the collection is empty.
- If the macAclInstanceNameIdCol collection contains any null element, or the collection contains an invalid MacAccessControlList InstanceNameId.

Parameters

opContext—Operational context.

vlanAccessMapEntryInstanceNameId—InstanceNameId of VlanAccessMapEntry object.

macAclInstanceNameIdCol—a collection of InstanceNameId of one or more MacAccessControlList objects.

Return Value

void

unbindNetworkInterfacesFromVlanAccessMapEntry

Clears the given list of network interfaces that are assigned as redirect interfaces, from the VACE.

ValidationException is thrown if any of the following situation occurs:

- If vlanAccessMapInstanceNameId is null or it is not of type VlanAccessMapEntry InstanceNameId.
- If vlanAccessMapInstanceNameId is not a valid VlanAccessMapEntry InstanceNameId.
- If the networkInterfaceInstanceNameIdCol collection is null or the collection is empty.
- If the networkInterfaceInstanceNameIdCol collection contains any null element, or the collection contains an invalid NetworkInterface InstanceNameId.
unbindTimeRangeFromAces

Clears timerange from one or more ACEs.

ValidationException is thrown if any of the following situation occurs:
- If the aceInstanceNameIdCol collection is null or the collection is empty.
- If the aceInstanceNameIdCol collection contains any null element, or the collection contains an invalid ExtendedAccessControlEntry, Ipv6AccessControlEntry or RoleBasedAccessControlEntry InstanceNameId.

Parameters
opContext—Operational context.
aceInstanceNameIdCol—a collection of InstanceNameId of one or more ACEs. ACEs can be ExtendedAccessControlEntry, Ipv6AccessControlEntry or RoleBasedAccessControlEntry.

Return Value
void

unbindVlanAccessMapFromVlans

Clears VACL from one or more VLANs.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameId is null or it is not of type NetworkElement InstanceNameId.
- If neInstanceNameId is not a valid NetworkElement InstanceNameId.
- If the vlanIds value is null.

Parameters
opContext—Operational context.
neInstanceNameId—InstanceNameId of a NetworkElement.
vlanIds—One or more VLAN IDs, that uniquely identifies a VLAN.

Return Value
void
CollectorApp Service

This chapter describes the DCNM web services’ API methods for the CollectorApp service.

Information About CollectorApp Service

The collector allows you to address performance management of network resources. By analyzing statistical data with the collector, you can perform real-time monitoring of data that includes the traffic throughput, the percentage utilization, error rates, and the response time. As a result, you can efficiently use the network resources and have an in-depth knowledge of the network health.

The collector can read statistical information from the devices and store it in the database. You can use the collector APIs to retrieve the information stored in the database.

To perform a collection of statistical data, follow these steps:

1. Create a CollectionInfo by specifying a collection information name, a list of statistical data, a schedule (simple or cron style), and a collection action (persist).
2. Start scheduling the collection information.

To delete an existing collection of statistical data, follow these steps:

1. Stop the CollectionInfo if it is started.
2. Delete the collection information.

addStatisticalEntry

Associates the given statistical entry to an existing collection info. The given statistical entry should not be duplicated within the collection info.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.
- If the aStatisticalEntry is null

CollectorException is thrown if any of the following situation occurs:

- If the aStatisticalEntry already exist for the given collection info
- If the aStatisticalEntry have a configInstanceId which is belongs to a different device than the given collection info
createCollector

Creates a collection info and returns its instance name id.
The given collection info object should have one or more statistical entries, a schedule and collector action. Collector will start polling as per the schedule given. If there is no start time is specified in the schedule, collector will start polling immediately.

ValidationException is thrown if any of the following situation occurs:
- If aCollectionInfo is null.

CollectorException is thrown if any of the following situation occurs:
- If name is not specified in aCollectionInfo.
- If collector action is not specified in aCollectionInfo.
- If schedule is not specified in aCollectionInfo.
- If statistical entries are empty in aCollectionInfo.
- If statistical entries are duplicated in aCollectionInfo.
  In the given statistical entries,
  - If statistical class name is null or configInstanceId is null or invalid.
  - If all the configInstanceIds are not belonging to the same device.
  - If there is no relation between statistical class and configInstanceId.
  In the given schedule,
  - If the expression is invalid for Cron schedule.
  - If the specified end time is ending before start time.
  - If the specified interval is less than 30 seconds.

Parameters
opContext—Operational context
aCollectionInfo—instance of CollectionInfo

Return Value
InstanceNameId the instance name id of the CollectionInfo
deleteCollectionInfo

Deletes an existing collection info and the collected statistical data for the given collection info.

Stops the collector if it is running for the given collection info.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

IntegrityException is thrown if the given collectionInfo does not exist in devices.

**Parameters**

- opContext—Operational context
- collectionInfoId—instance id of a CollectionInfo.

**Return Value**

void

deleteStatisticalData

Deletes the collected statistical data for the given collection info.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

IntegrityException is thrown if the given collectionInfo does not exist in devices.

**Parameters**

- opContext—Operational context
- collectionInfoId—instance id of a CollectionInfo.

**Return Value**

void

deleteStatisticalDataTillDate

Deletes the collected statistical data for the given collection info.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

IntegrityException is thrown if the given collectionInfo does not exist in devices.

**Parameters**

- opContext—Operational context
getAllCollectionInfo

Returns all existing collection info which matches with given statisticalclassname, configInstanceId and interval. If none of them are specified, returns all the CollectionInfos.

Parameters
opContext—Operational context
statisticalClassName—statistics class name
configInstanceId—InstanceNameId of the configuration instance
interval—polling interval

Return Value
All the CollectionInfo which matches with the given parameters

g getAllCollectionInfoByConfigIds

Returns the active collection info which exactly matches with given statisticalclassname, configInstanceIds and interval. If none of them are specified, returns null.

Parameters
opContext—Operational context
statisticalClassName—statistics class name
configInstanceIds—List of InstanceNameId of the configuration instances
interval—polling interval

Return Value
Only one CollectionInfo which matches with the given parameters

g getAllCollectionInfoByStatus

Returns all existing collection info with a specific collection status. ValidationException is thrown if any of the following situation occurs:
- If aCollectionStatus is null.

Parameters
opContext—Operational context
aCollectionStatus—status of the CollectionInfo
getAllCollectionInfoByUserId

Returns all existing collection info created by a user.
ValidationException is thrown if any of the following situation occurs:
- If userId is null.

Parameters
- opContext—Operational context
- username—name of the user who created the CollectionInfo.

Return Value
All the CollectionInfo created by the specified username.

getAllCollectionInfoInDevice

Returns all existing collection info bound to a network element.
ValidationException is thrown if any of the following situation occurs:
- If neInstanceId is null.
- If neInstanceId is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- networkElementId—instance id of the network element.

Return Value
All the CollectionInfo having the specified CollectionStatus.

getAllCollectionInfoInDeviceByStatus

Returns all existing collection info bound to a network element and having a specific collection status.
ValidationException is thrown if any of the following situation occurs:
- If neInstanceId is null.
- If neInstanceId is not a valid network element InstanceNameId.
- If aCollectionStatus is null.

Parameters
- opContext—Operational context
- aCollectionStatus—status of the CollectionInfo
getAllStatisticalEntriesForDevice

Returns all the statistical entries bound to a network element.
ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
networkElementId—instance id of the CollectionInfo

Return Value
All the statistical entries that were polled for the specified deviceId.

collectedData

Returns the last polled statistical data of an existing collection info with some additional information like next polling time.
ValidationException is thrown if any of the following situation occurs:
- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

Parameters
opContext—Operational context
collectionInfoId—instance id of the CollectionInfo

Return Value
Return an instance of ExtendedCollectedData which is the statistical data that is polled at the last. This is useful for real-time monitoring.

collectedStatisticalData

Deprecated.
Returns all the collected statistical data of an existing collection info.
ValidationException is thrown if any of the following situation occurs:
- If collectionInfoId is null.
getCollectionInfo

Returns the instance of collection info specified by collection info name id.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

Parameters

- opContext—Operational context
- collectionInfoId—instance name id of the CollectionInfo

Return Value

CollectionInfo CollectionInfo

getLatestCollectedStatisticalData

Returns the last polled statistical data of an existing collection info.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

Parameters

- opContext—Operational context
- collectionInfoId—instance id of the CollectionInfo

Return Value

Return an instance of CollectedData which is the statistical data that is polled at the last. This is useful for real-time monitoring.

getValidConfigInstanceId

Returns valid configuration instance name ids for the given configuration instance name ids.
Instancename ids will differ by the following cases, 1. Rediscovery 2. Type change ex. port mode change in interfaces 3. Data removed from the device 4. Identity is changed by APIs, ex: ACEs are recreated when ACL is getting updated through APIs. Valid instance name ids are returned by reloading from database for the given old instance name ids. ValidationException is thrown if any of the following situation occurs:

- If configInstanceIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId
- if the argument passed is null or it is not a valid InstanceNameId.

**Parameters**

- `opContext`—Operational context
- `configInstanceIdCol`—InstanceId of the one or more configuration entities.

**Return Value**

The returned list will contain valid InstanceNameIds.

### modifyCollectorAction

Modifies the collector action of an existing collection info. ValidationException is thrown if any of the following situation occurs:

- If `collectionInfoId` is null.
- If `collectionInfoId` is not a valid collection info InstanceNameId.
- If `aCollectorAction` is null.

**Parameters**

- `opContext`—Operational context
- `collectionInfoId`—instance id of the CollectionInfo.

**Return Value**

`void`

### modifySchedule

Modifies the schedule of an existing collection info. ValidationException is thrown if any of the following situation occurs:

- If `collectionInfoId` is null.
- If `aSchedule` is null.
- If `collectionInfoId` is not a valid collection info InstanceNameId.

In the given schedule,

- If the expression is invalid for Cron schedule.
- If the specified end time is ending before start time.
- If the specified interval is less than 30 seconds.
removeStatisticalEntry

Removes a StatisticalEntry from an existing collection info.
ValidationException is thrown if any of the following situation occurs:
- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.
- If the statisticalEntryId is null
- If statisticalEntryId is not a valid statistical entry InstanceNameId.
CollectorException is thrown if any of the following situation occurs:
- If statisticalEntryId is not exist within the given collection info
- If the given collection info is having only one statistical entry.

Parameters
opContext—Operational context
collectionInfoId—instance id of the CollectionInfo
statisticalEntryId—instance name if of a StatisticalEntry that needs to be added to CollectionInfo

Return Value
void

startCollector

Starts polling for an existing collection info with its schedule.
ValidationException is thrown if any of the following situation occurs:
- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.
- If aCollectorAction is null.

Parameters
collectionInfoId—instance id of the collection info
Send document comments to nexus7k-docfeedback@cisco.com

Return Value
void

startCollectorBySchedule

Starts polling for an existing collection info as per the given schedule. ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.
- If schedule is null.

In the given schedule,

- If the expression is invalid for Cron schedule.
- If the specified end time is ending before start time.
- If the specified interval is less than 30 seconds.

Parameters
collectionInfoId—instance id of the collection info

Return Value
void

stopCollector

Stops polling for an existing collection info. ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.
- If schedule is null.

Parameters
collectionInfoId—instance id of the collection info

Return Value
void
CollectorExtendedApp Service

This chapter describes the DCNM web services’ API methods for the CollectorExtendedApp service.

Information About CollectorExtendedApp Service

The collector allows you to address performance management of network resources. By analyzing statistical data with the collector, you can perform real-time monitoring of data that includes the traffic throughput, the percentage utilization, error rates, and the response time. As a result, you can efficiently use the network resources and have an in-depth knowledge of the network health.

The collector can read statistical information from the devices and store it in the database. You can use the collector APIs to retrieve the information stored in the database. In addition, you can use the extended collector API to trend statistical data over a period of time.

getCollectedStatisticalData

Returns all the collected statistical data of an existing collection info.
ValidationException is thrown if any of the following situation occurs:
• If collectionInfoId is null.
• If collectionInfoId is not a valid collection info InstanceNameId.

Parameters
opContext—Operational context
collectionInfoId—instance id of the CollectionInfo

Return Value
Return a collection of CollectedData which is all the statistical data that is polled for the specified collectionInfoId.

getCollectedStatisticalDataByDaysMaxLimit

Returns List of collected statistical data of an existing collection info during a time period specified by days. Filter the result by given max limits.
getCollectedStatisticalDataByMaxLimit

Returns List of collected statistical data of an existing collection info and filter the result for the max limits.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

Parameters

- opContext—Operational context
- collectionInfoId—instance id of the CollectionInfo
- maxLimits—Specifies the max points to be returned

Return Value

Return List of the collection of CollectedData which is all the statistical data that is polled for the specified collectionInfoId within the max points.

getCollectedStatisticalDataByTimeRange

Returns all the collected statistical data of an existing collection info during a time period specified by the start time and end time.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

Parameters

- opContext—Operational context
- collectionInfoId—instance id of the CollectionInfo
getCollectedStatisticalDataByTimeInterval

Returns all the collected statistical data of an existing collection info during a time period specified by the start time and end time. Filter the result by given interval.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

Parameters

- opContext—Operational context
- collectionInfoId—instance id of the CollectionInfo
- startTime—Specifies the start of the time range to get collected statistical data.
- endTime—Specifies the end of the time range to get collected statistical data.
- intervalInSeconds—Specifies the interval needs to be filtered

Return Value

Return a collection of CollectedData which is all the statistical data that is polled for the specified collectionInfoId during the specified time range and the interval.

getCollectedStatisticalDataByTimeIntervalMaxLimit

Returns all the collected statistical data of an existing collection info during a time period specified by the start time and end time. Filter the result by given max limits.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

Parameters

- opContext—Operational context
- collectionInfoId—instance id of the CollectionInfo
- startTime—Specifies the start of the time range to get collected statistical data.
- endTime—Specifies the end of the time range to get collected statistical data.
- maxLimits—Specifies the max points to be returned
getCollectedStatisticalDataSizeByTimeRangeInterval

Returns size of the collected statistical data of an existing collection info during a time period specified by the start time and end time. Filter the result by given interval.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

Parameters

- opContext—Operational context
- collectionInfoId—instance id of the CollectionInfo
- startTime—Specifies the start of the time range to get collected statistical data.
- endTime—Specifies the end of the time range to get collected statistical data.
- intervalInSeconds—Specifies the interval needs to be filtered

Return Value

Return size of the collection of CollectedData which is all the statistical data that is polled for the specified collectionInfoId during the specified time range and the interval.

def getPreviousCollectedData

Returns the previous to the last polled statistical data of an existing collection info.

ValidationException is thrown if any of the following situation occurs:

- If collectionInfoId is null.
- If collectionInfoId is not a valid collection info InstanceNameId.

Parameters

- opContext—Operational context
- collectionInfoId—instance id of the CollectionInfo

Return Value

Return an instance of CollectedData which is the statistical data that is previous to the last polled data.
ConfigArchiveApp Service

This chapter describes the DCNM web services’ API methods for the ConfigArchiveApp service.

Information About ConfigArchiveApp Service

The ConfigArchive component allows you to address the configuration archival and restore operation on each device. The purpose of the configuration archiver is to periodically poll the monitored NX-OS devices in a network and archive their running and startup configurations. As a result, it is possible to reload a prior configuration if it has been saved. If the configuration has not been saved, you will have to recreate or reconstruct the configuration before you restore the service.

The ConfigArchive application allows you to do the following:

- Track all configuration changes for a device at a scheduled interval or time.
- Track when the configuration changes.
- Retrieve the configuration archival history.
- Roll back the history or roll back to a specific version.
- Schedule the time for the configuration archival.
- Obtain the raw configuration data of specific versions for particular devices.

createConfigArchiveTask

This API takes schedule for the config archival creates and starts the Config Archival Task according to schedule. The config Task include the list of device Instance Name Id's to be included in the scheduling task. Schedule can be either periodic schedule or Cron Schedule.

Parameters

- opContext—Operational context
- List—is the list of devices to be included in the config archival task.
- schedule—is the schedule which can be a periodic schedule or a chronic schedule. The config archival service starts the archival process as per the schedule.

Return Value

void
deleteConfigArchiveStatusHistoryTillDateForTask

This API is used to delete the config archive status history till the given Date for the given task.

Parameters
opContext—Operational context
instanceId—is the InstanceNameId of the config archive task.
timestamp—is the timestamp till what user wants to delete.

Return Value
void

deleteConfigArchiveTasks

This API is used to delete the existing config archive tasks.

Parameters
opContext—Operational context
InstanceIdList—is the List of instanceIds of the tasks that needs to be deleted.

Return Value
void

deleteConfigFilesTillDate

This API is used to delete the config files till the given Date. Both the archived config files as well as database metadata will be deleted.

Parameters
opContext—Operational context
vdcInstanceIdList—is the list of vdc instanceid's for which the configversion files needs to be deleted.
timestamp—is the timestamp till what user wants to delete.

Return Value
void

deleteRollbackStatusHistoryTillDateForDevice

This API is used to delete Rollback Status history till the given Date for the given list of vdc's

Parameters
opContext—Operational context
timestamp—is the timestamp till what user wants to delete.
fetchCfgDiff

This API is used to retrieve the diffConfig commands between any two configuration files across archived configuration files, live(Startup/running) configuration data. The API first fetches the configuration file of the left device, if lVer is null, it fetches the live data else appropriate configuration file version is fetched. Similarly the API fetches for the second configuration file from the device or the configuration version.

Parameters
opContext—Operational context
lvdcInstanceId—Left selected device instance name id.
lVer—Version of the configuration file for the left selected device. if it is null, the live data is considered.
lType—Whether the configuration file is of STARTUP or RUNNING. if it is null, RUNNING config is considered for the left selected device.
rvdcInstanceId—Right selected device instance name id.
rVer—Version of the configuration file for the right selected device. if it is null, the live data is considered.
rType—Whether the configuration file is of STARTUP or RUNNING. if it is null, RUNNING config is considered for the right selected device.

Return Value
Returns the collection of Configuration-command-object-collection of given left file version(lVer) and right file version(rVer), where the configuration-command-object contains the information of the mapped configuration-command-object on the either version file.

fetchNetworkElementConfigData

This API is used to get the current configuration data for the given list of vdc's This config data will not be archived.

Parameters
opContext—is the opcontext.
vdcInstanceIdList—is the list of instanceNameld's of vdc's for which config data needs to be fetched.
type—
getAllConfigTasks

This API gets all the config archive tasks that are created in the database. The config archive task can be either in active state or in inactive state. Each config archive task may involve one or more devices.

Parameters
- opContext—Operational context

Return Value

getArchivalDeviceStatusEntries

This API is used to get the archiv al status information for each device given the instanceNameId of the device.

Parameters
- vdcInstanceIdList—is the collection of instanceNameIds of the devices.

Return Value
- Returns the collection of DeviceStatusEntries collection for each device instanceNameId.

get ArchivedNetworkElementConfigData

This API is used to get the actual raw data of the configuration. It takes a list of vdc InstanceId's and list of versions and returns the list of ConfigData. ConfigData will have the data in compressed form, getText() method in it returns the uncompressed form. There will be one to one mapping between vdc instanceId and the version.

Parameters
- opContext—Operational context
- vdcInstanceIdList—is the list of vdc instanceId's for which config data needs to be fetched.
- versionList—is the list of versions mapping to the vdc.

Return Value

getConfigArchiveSettings

This API is used to get Global Config Archive Settings. Config Archive Settings include the Max Version Count to return and max status history count to return.

Parameters
- opContext—Operational context
Chapter 6      ConfigArchiveApp Service

Send document comments to nexus7k-docfeedback@cisco.com

Return Value
ConfigArchiveSettings which include statusHistoryCount and maxVersionCount

getConfigArchiveStatusHistoryForTask

The API is used to get the ConfigArchiveStatus History for the given task. from the database.

Parameters
opContext—Operational context
InstanceNameId.—is the InstanceNameId of task for which the config archive status history needs to be fetched.

Return Value
List is the list of ConfigFetchHistory type. ConfigFetchHistory will contain information about config archival operation,timestamp and the list of devices.

getConfigVersions

This API is used to fetch the ConfigHistoryMetadata for the list of vdc instance ids passed. It also takes config type as the parameter to check if the user has requested for Startup Config or Running Config

Parameters
opContext—Operational context
vdcInstanceId—vdcInstanceIdList is the instanceNameId's of the vdc's for which user is fetching the config metadata.
type—is the type of the configuration. Config type can be Running or Starting.

Return Value
List returns a list of meta data of each config version.

getRollbackStatusHistoryForDevice

This API is used to get the RollbackStatusHistory for the given devices from the database.

Parameters
opContext—Operational context
vdcInstanceIdList—is the list of instanceNameIds of the vdc's for which roll back status history needs to be fetched.

Return Value
List is the list of RollbackHistory type. RollbackHistory will contain information about the device instanceNameId for which the roll back has been performed. timestamp, result and the description of the result.
modifyConfigArchiveSettings

This API is used to modify the values of Config Archive Settings.

Parameters

- opContext—Operational context
- setting—is the modified Config Archive Setting.

Return Value

- void

modifyConfigArchiveTask

This API is used to modify the existing Config Archival Task. User can add new devices or delete existing devices to the existing task or he can modify the existing modified schedule.

Parameters

- opContext—Operational context
- taskId—is the taskId for which user which user wants to modify the schedule and devices
- deviceIdList—is the new device Id List.
- sch—is the new schedule

Return Value

- void

modifyConfigVersion

This API is used to modify the existing config versions with the new comment in the Pojo.

Parameters

- opContext—Operational context
- configVersionList—is the list of configVersions that needs to be modified.

Return Value

- void

performConfigArchive

This API is used to perform the on demand config archive operation for the given list of vdc’s.

Parameters

- opContext—Operational context
**performRollback**

This API takes the version of the config and the type for a particular vdc Instance Id and performs the roll back operation on the device.

**Parameters**
- **opContext**—Operational context
- **vdcInstanceId**—is the instanceNameId of the vdc for which user is doing rollback.
- **type**—is the type of the configuration. Config type can be Running or Starting.
- **Version**—is the version of the config file to which the user is performing the roll back operation. *
- **@param saveToStartup** is the boolean indicating whether the running configuration needs to be save to startup configurations.

**Return Value**
void

**startConfigArchiveTask**

This API is used to start the config archive task which is suspened earlier. If the VdcInstanceNameId is null, it will restart the task for all network elements involved in the task. Else only the job for a given vdc will be started.

**Parameters**
- **opContext**—is the opcontext
- **taskInstanceNameId**—is the instanceNameId of the task which needs to be started
- **vdcInstanceNameId**—is the VdcInstanceNameId for which the job needs to started

**Return Value**
void

**stopConfigArchiveTask**

This API is used to stop the config archive task which is running. If the VdcInstanceNameId is null, it will stop the task for all network elements involved in the task. Else only the job for a given vdc will be stopped.
Parameters

opContext—is the opcontext

taskInstanceNameId—is the instanceNameId of the task which needs to be stopped

vdcInstanceNameId—is the VdcInstanceNameId for which the job needs to stopped

Return Value

void
DaiApp Service

This chapter describes the DCNM web services’ API methods for the DaiApp service.

Information About DaiApp Service

Dynamic Address Resolution Protocol (ARP) inspection (DAI) is a security feature that validates ARP packets in a network. DAI intercepts, logs, and discards ARP packets with invalid IP-to-MAC address bindings, which allows you to protect the network from man-in-the-middle attacks, where an attacker could send forged ARP packets (for example, gratuitous ARPs) carrying a bogus IP/MAC binding in the payload to a host or to the default gateway.

bindArpAclOnVlans

Applies the ARP ACL on to a collection of VLANs. API to validate whether the VLANs already has an ARP ACL. API shall also validate whether DAI is enabled on the VLAN. If DAI is enabled on the VLAN, then API shall throw a warning message stating that ARP inspection based on ARP ACL will take precedence over validation done by DAI. That is, if a packet has to be denied based on ARP ACL, it will be denied even though the packet is valid as per DAI.

ValidationException is thrown if any of the following situation occurs:

- If arpAclInstanceNameId is null or it is not of type ARP ACL InstanceNameId.
- If the vlanInstanceNameIdCol is null or empty.
- If the vlanInstanceNameIdCol collection contains one null element, or the collection contains an invalid VLAN InstanceNameId.

Parameters

- opContext— Operational context.
- arpAclInstanceNameId— InstanceNameId of ARP ACL object.
- vlanInstanceNameIdCol— A collection of InstanceNameId of VLAN.
- explicitDenyEnable—Indicates whether the ARP ACL has to be configured as a static ACL.

Return Value

void
bindArpAclOnVlansByArpAclName

Binds pre-provisioned ARP ACL to a collection of VLANs. For ARP ACL to be bound to a collection of VLANs, ARP ACL need not be physically configured in the device. We can bind ARP ACL to a collection just by using the name of ARP ACL. This API addresses this pre-provisioning configuration.

ValidationException is thrown if any of the following situation occurs:

- If the passed argument arpAclName is null.
- If the vlanInstanceIds collection is null or empty.
- If the vlanInstanceIds collection one null element, or the collection contains an invalid VLAN instance name Id.
- If the vlanInstanceIds collection contains an instance ID of a VLAN that does not exist in the database.

PropertiesException is thrown if any of the following situation occurs.

- If arpAclName contains an invalid ARP ACL name string.

Example:

- If the ARP ACL name does not start with an alphabet like "2acl_test".
- If the ARP ACL name contains a space or question mark or quotation mark character like "acl test2" or "acl?test2" or "acl\'test".

Parameters

- opContext—— Operational context.
- arpAclName—— Name of the ARP ACL. This ACL need not be configured in the device.
- vlanInstanceIds—— A collection of InstanceNameId of VLANs in a network element that are configured in the device.
- explicitDenyEnable—— Indicates whether the ARP ACL has to be configured as a static ACL.

Return Value

- void

bindArpAclOnVlansForRange

Binds pre-provisioned ARP ACL to a collection of pre-provisioned VLANs. For ARP ACL to be bound to a collection of VLANs, both ARP ACL and VLAN need not have been configured in the device. Users can bind ARP ACL name to a collection of VLANs and then create both ARP ACLs and the bound VLANs at a latter stage. This API addresses this pre-provisioning configuration.

ValidationException is thrown if any of the following situation occurs:

- If the passed argument arpAclName is null.
- If the passed argument vlanRange is null.
- If the networkElementId is null or not of type network element InstanceNameId.
- If the network element with the InstanceNameId given by networkElementId does not exist in the database.

PropertiesException is thrown if any of the following situation occurs:
clearArpRateLimitingConfigurationInInterfaces

Clears the ARP rate limiting and burst interval configurations done in a collection of interfaces. This API also restores the rate limiting and burst interval values to default values.

ValidationException is thrown if any of the following situation occurs:

- If the argument interfaceNameIds is null or it is not of type interface InstanceNameId.
- If the interface specified by any of the InstanceNameId in the collection interfaceNameIds does not exist in the database.

Parameters

opContext— Operational context.
interfaceNameIds— A collection of InstanceNameId of the interfaces.

Return Value

void

clearArpRateLimitingConfigurationInInterfaces

createArpAcls

Creates one or more standard ARP ACL objects in a network element. Given the InstanceNameId of a network element and a list of ARP ACL objects, creates the objects in the server and returns the collection of InstanceNameId of ARP ACLs created.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null or it is not of type network element InstanceNameId.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the ARP ACLs objects in the arpAclCol do not have their name attribute set.

Parameters

opContext— Operational context.
networkElementId— InstanceNameId of the network element.
arpAclName— Name of the ARP ACL. This ACL need not have been configured in the device.
vlanRange— A String representing the range of VLANs. The string holds comma separated / hyphenated list of VLANs. For example, vlanRange could be 4,6,9,15-20,25.
explicitDenyEnable— Indicates whether the ARP ACL has to be configured as a static ACL.

Return Value

void

createArpAcls
deleteArpAcls

If the arpAclCol contains duplicate entries of the ARP ACLs.
java.xml.bind.PropertyException is thrown if any of the following situation occurs:
• If name of an ARP ACL does not start with alphabets.
• If name of an ARP ACL contains space or quotation mark character.
• If name of an ARP ACL contains more than 234 characters.

IntegrityException is thrown if any of the following situation occurs:
• If the arpAclCol contains an ARP ACL that already exists in the database.
• If an ARP ACL in the arpAclCol contains duplicate ARP ACL entry objects.

VlanExternal associations with the ARP ACLs will not be considered by this API. User has to call separate API to bind the ARP ACL to a VLAN.

Parameters
opContext— Operational context.
networkElementId— InstanceNameId of the network element.
arpaclCol— A collection of ARP ACL objects to be created.

Return Value
A collection of newly created ARP ACL objects.

deleteArpAcls

Deletes one or more ARP ACLs. Given the InstanceNameId of ARP ACL objects, those objects will be deleted from the server.
ValidationException is thrown if any of the following situation occurs:
• If the arpAcls collection is null or empty.
• If the arpAcls collection contains an element that is not of type ARP ACL.
• If the arpAcls collection contains an ARP ACL that does not exist in the database.

Parameters
opContext— Operational context.
arpaclInstanceNameIds— A collection that contains InstanceNameId of ARP ACLs to be deleted

Return Value
void

disableDaiOnVlans

Disables DAI for a given collection of VLANs in a network element.
ValidationException is thrown if the argument passed is null or it is not of type VLAN InstanceNameId.
**Chapter 7      DaiApp Service**

---

### enableDaiOnVlans

Enables DAI for a given collection of VLANs in a network element. API to validate whether DHCP Snooping is enabled on the VLAN. If not, API to throw an Exception stating that DHCP Snooping has to be enabled for DAI to function.

ValidationException is thrown if the argument passed is null or it is not of type VLAN InstanceNameId.

#### Parameters

- `opContext`—— Operational context.
- `vlanIds`—— InstanceNameId of the VLANs.

#### Return Value

void

### enableDaiOnVlansByRange

Enables Dynamic ARP Inspection in a pre-provisioned VLAN. It is possible to enable Dynamic ARP Inspection just by using VLAN IDs. The VLANs in which Dynamic ARP Inspection has to be enabled need not actually exist in the device. This API addresses this pre-provisioning configuration.

ValidationException is thrown if any of the following situation occurs.

- If the passed argument `networkElementId` is null or not of type network element InstanceNameId.
- If the passed argument `vlanRange` is null.

#### Parameters

- `opContext`—— Operational context.
- `networkElementId`—— InstanceNameId of the network element.
- `vlanRange`—— A String representing the range of VLANs. The string holds comma separated / hyphenated list of VLANs. For example, `vlanRange` could be `4,6,9,15-20,25`.

#### Return Value

void
getArpAclsInNetworkElement

Returns a collection of all ARP ACLs configured in a given network element, given the InstanceNameId of the network element.

ValidationException is thrown if the argument passed is null or it is not of type network element InstanceNameId.

Parameters
- opContext— Operational context.
- networkElementId— InstanceNameId the network element.

Return Value
List of all ARP ACL objects present in the network element. In the returned list of objects, only the following associations will be present and other associations will be cleared:
- All associated ACEs of the returned ARP ACL object.
- The VLANs which are referring the returned ARP ACL object.
- The network element in which the returned ARP ACLs are configured.

getArpAclsInVlans

Given a collection of VLAN InstanceNameIds, this API returns the ARP ACLs associated to it, if any.

Parameters
- opContext—
- vlanInstanceNameIdCol—

Return Value
List of ARP ACLs associated to VLAN. If no ARP ACL is associated to a VLAN in the parameter collection, then the list will contain null element in the corresponding position.

getArpAclsWithoutAcesInNetworkElement

Returns a collection of all ARP ACLs configured in a given network element, given the InstanceNameId of the network element.

ValidationException is thrown if the argument passed is null or it is not of type network element InstanceNameId.

Parameters
- opContext— Operational context.
- networkElementId— InstanceNameId the network element.

Return Value
List of all ARP ACL objects present in the network element. In the returned list of objects, all associations will be cleared.
getDaiDisabledVlansInNetworkElement

Returns all the DAI Disabled VLANs in the given network element.
ValidationException is thrown if the argument passed is null or it is not of type network element InstanceNameId.

Parameters
opContext— Operational context.
networkElementId— InstanceNameId of the network element.

Return Value
A list of DAI disabled VlanExternal objects. In the returned list of objects, only the following associations will be present and other associations will be cleared.

- DaiSetting association.
- ARP ACL association. If that ARP ACL has ARP ACL entries, those associations will be cleared.

getDaiEnabledVlansInNetworkElement

Returns all the DAI Enabled VLANs in the given network element.
ValidationException is thrown if the argument passed is null or it is not of type network element InstanceNameId.

Parameters
opContext— Operational context.
networkElementId— InstanceNameId of the network element.

Return Value
A list of DAI enabled VlanExternal objects. In the returned list of objects, only the following associations will be present and other associations will be cleared.

- DaiSetting association.
- ARP ACL association. If that ARP ACL has ARP ACL entries, those associations will be cleared.

getDaiGlobalSettingsInNetworkElements

Gets the DAI global settings for a given list of network elements. Given the InstanceNameId of the network element, returns a collection of DAI global settings.
ValidationException is thrown if the passed argument is null or not of type network element InstanceNameId.

Parameters
opContext— Operational context.
neInstanceId—— InstanceNameId of the network elements.
getDaiSettingOnVlans

Returns the list of DAI settings pertaining to the given list of VLANs.
ValidationException is thrown if the argument passed is null or it is not of type VLAN InstanceNameId.

Parameters
opContext—- Operational context.
vlans—— InstanceNameId of the VLANs.

Return Value
A list of DaiSetting objects. In the returned list of objects, only the following associations will be present and other associations will be cleared.

- VLAN association.
- ARP ACL associated with the VLAN, that is associated with the DAI Setting. If that ARP ACL has ARP ACL entries, those associations will be cleared.

getInterfacesWithArpRateLimitingInNetworkElement

Returns all the interfaces in the given network element having ARP rate and burst interval configured. These interfaces have configured values for ARP rate limiting and burst interval.
ValidationException is thrown if the argument passed is null or it is not of type network element InstanceNameId.

Parameters
opContext—- Operational context.
networkElementId—— InstanceNameId of the network element.

Return Value
A collection of interface objects having ARP rate limiting and burst interval configured.

getTrustStateSettingInInterfaces

Returns the collection of trust state setting objects. Given the collection of interface InstanceNameId, returns a collection of trust state setting objects.
ValidationException is thrown if the argument passed is null or it is not of type interlace InstanceNameId.

Parameters
opContext—- Operational context.
interfaceInstanceIds—- InstanceNameId of interfaces.

Return Value
A collection of TrustStateSetting objects.

getUntrustedInterfacesWithDefaultRateInNetworkElement

Returns all the untrusted interfaces in the given network element having default ARP rate and burst interval values, given the InstanceNameId of the network element.

ValidationException is thrown if the argument passed is null or it is not of type network element InstanceNameId.

Parameters
opContext—- Operational context.

networkElementId—- InstanceNameId of the network element.

Return Value
A collection of untrusted interface objects having default rate limiting and burst interval.

getVlansWithDaiSettingNetworkElement

Returns all the VLANs with DaiSetting object, in the given network element.

ValidationException is thrown if the argument passed is null or it is not of type network element InstanceNameId.

Parameters
opContext—- Operational context.

networkElementId—- InstanceNameId of the network element.

Return Value
A list of VLANs with DaiSetting object.In the returned list of objects, only the following associations will be present and other associations will be cleared.

-  DaiSetting association.
-  ARP ACL association. If that ARP ACL has ARP ACL entries, those associations will be cleared.

modifyAclSequence

Modifies the sequence number of the ACEs in an ACL, based on the starting sequence number and the step to increment the sequence numbers.

Parameters
opContext—- Operational context.

aclInstanceNameIdCol—- InstanceNameId of one or more ACLs.
modifyArpAcls

Modifies one or more existing ARP ACL objects.

ValidationException is thrown if any of the following situation occurs:

- If arpAcls collection is null or empty.
- If arpAcls collection contains an object that is not of type ARP ACL.

PropertiesException is thrown if any of the following situation occurs:

- If the name of the ARP ACL in the arpAcls collection is modified.

IntegrityException is thrown if any of the following situation occurs:

- If the arpAcls collection contains an ARP ACL object that does not exist in the database.
- If the ARP ACL in the arpAcls collection contains a duplicate ARP ACL entry objects.

This API will not consider VLAN association of the ARP ACL objects. User needs to call separate API to bind an ARP ACL to a VLAN.

Parameters

opContext—— Operational context.

arpAcls—— a collection of modified ARP ACL objects that will replace the existing ARP ACL objects in the database.

Return Value

Void.

modifyDaiGlobalSettingsInNetworkElements

Modifies one or more existing DAI global setting objects in a given collection of network elements.

ValidationException is thrown if any of the following situation occurs:

- If the networkElementIds collection is null or empty.
- If the networkElementIds collection contains one null element, or the collection contains an invalid network element InstanceNameId.

PropertiesException is thrown if any of the following situation occurs:

- In the daiGlobalSettings collection, if any one of the daiGlobalSetting attribute is not valid.

Example:

- If logBufferSize attribute does not contains a value between 0—1024.
modifyDaiOnVlans

Modifies one or more existing DAI setting objects.
ValidationException is thrown if any of the following situation occurs:
- If modifiedDaiSettings collection is null or empty.
- If modifiedDaiSettings collection contains an object that is not of type DaiSetting.

Parameters
opContext—— Operational context.
modifiedDaiSettings—— A collection of modified DaiSetting objects that will replace the existing DAI setting objects in the database.

Return Value
void

modifyDaiSettingsAndArpAclBindingsOnVlans

Modifies DAI settings and ARP ACL bindings in a given collection of VLANs. VLAN objects passed shall have modified DAI settings and modified ARP ACL bindings.
Modification of ARP ACL Entry is not supported in this API. use modifyArpAcls(List arpAcls) API to modify ARP ACL Entry of an ARP ACL.
ValidationException is thrown if any of the following situation occurs:
- If modifiedVlanObjects collection is null or empty.
- If the modifiedVlanObjects contains a VLAN that does not exist in the database.
- If the VLAN in the database, corresponding to the elements in modifiedVlanObjects collection does not contain DAI setting.
modifyTrustStateSettings

Modifies one or more existing trust state setting objects in a given collection of network interfaces.

PropertiesException is thrown if any of the following situation occurs:
- If interfaceInstanceIds collection is null or empty.
- If interfaceInstanceIds collection contains one null element, or the collection contains an invalid interface InstanceNameId.
- If the trustStateSettings collection is null or empty.
- If the trustStateSettings collection contains an element that is not of type TrustStateSetting.

PropertiesException is thrown if any of the following situation occurs:
- If the arpRate attribute does not contain a value between 0—2048.
- If the burstInterval attribute does not contain a value between 0—15.

IntegrityException is thrown if any of the following situation occurs:
- If the interfaceInstanceIds and trustStateSettings collections size are not equal.
- If the trustStateSettings collection contains an interface which is not of type TrustStateSetting.

Parameters
- opContext— Operational context.
- interfaceInstanceIds— InstanceNameIds of interfaces.
- trustStateSettings— Modified TrustStateSetting objects.

Return Value
void

unbindArpAclFromVlans

Removes the association with ARP ACLs in a collection of VLANs

ValidationException is thrown if any of the following situation occurs:
If the `vlanInstanceNameIds` collection is null or empty.

- If the `vlanInstanceNameIdCol` collection contains one null element, or the collection contains an invalid VLAN instance name ID.

**Parameters**

- `opContext`— Operational context.
- `vlanInstanceNameIds`— InstanceNameId of VLANs in which the ARP ACL association has to be removed.

**Return Value**

`void`
DataPurgingApp Service

This chapter describes the DCNM web services’ API methods for the DataPurgingApp service.

Information About DataPurgingApp Service

When the system database gets too large, it can slow resources, corrupt databases, and fill available disk space. The DataPurging application allows you to purge the system of all call records that are no longer necessary and to delete or consolidate the entries in the database.

createDataPurgingSchedule

Creates the datapurging schedule with the given schedule and threshold information. If the purgingEnable attribute is set to TRUE then a timer task(scheduler) is created, the corresponding purging handler is assigned to that timer task and the schedule is scheduled to run. Otherwise, the schedule is just persisted in the database. A Map of schedulerType and the PurgingHandler (com.cisco.dcbu.dcm.scheduler.statistics.archive.PurgingHandler) is maintained. Whenever a new purging schedule is created, an entry will be put into that map.

ParameterException is thrown if the following situation occurs

- If the schedule which is passed as a parameter to this API is null.
- If the schedulerType SchedulerType is null.
- If both, the scheduleDays is null or empty and the daily attribute is false or empty, in the schedule.

MetadataException is thrown if the following situation occurs.

- If the value for the scheduleAt attribute (the time at which the purging has to happen) in the scheduled days is null

InstantiationException is thrown if there exist a datapurging schedule in the database, corresponding to the schedulerType.

Parameters

opContext—Operational context

schedule—DataPurgingSchedule It contains the information such when the schedule has to be run, what kind of purging action PurgingAction has to be taken on Purging, which data (scheduler tye - SchedulerType) has to be purged and which data has to be purged.

There are two attributes that state which data has to be purged and which has to be left unprocessed.
**deleteDataPurgingSchedule**

Deletes the datapurging schedule corresponding to the given schedulerType. Before deleting the schedule from the database, this API first cancels the corresponding scheduler (timer task) and its task (purging handler). If there is no schedule exists in the database for the given schedulerType, calling this API does not have any impact. ParameterException is thrown if the schedulerType which is passed as a parameter is null.

**Parameters**
- opContext—Operational context
- type—Scheduler type

**Return Value**
void

**getDataPurgingSchedule**

Returns the Data purging schedule corresponds to the scheduler type. ParameterException is thrown if the argument passed is null.

**Parameters**
- opContext—Operational context
- schedulerType—The type of the data for which purging has to be done 1. EVEN 2. COLLECTOR

**Return Value**
The Data purging schedule corresponding to the scheduler type DataPurgingSchedule objects.

**modifyDataPurgingSchedules**

Modifies the purging schedule with the given schedule which is passed as an argument. A new timer task is created, assigned with the corresponding purging handler task and the old timer is cancelled and removed.

ParameterException is thrown if the following situation occurs
- If the schedule which is passed as a parameter to this API is null.
- If the schedulerType SchedulerType is null, in the schedule DataPurgingSchedule or if there is no schedule present in the DB corresponding to the schedulerType set in the schedule.
purgeDataOnDemand

Purges the data immediately. There are two ways to purge the data. The first one is auto purging, which runs based on the schedule. The second one is manual purging, where the user can purge the data whenever he wants. But this purging happens only once upon user request, and does not run on schedule basis. The threshold to keep the raw data will be taken from the schedule object which is passed as an argument. If the user has modified the threshold and called this API, those thresholds will not be persisted. They are just taken as a floating data and used for the purging which will happen on this API call.

ParameterException is thrown if the following situation occurs

- If the schedule which is passed as a parameter to this API is null.
- If the schedulerType SchedulerType is null.
- If the value for both skipRowCount and skipPastDaysCount attribute is null.
- If the value for skipRowCount or skipPastDays count is negative (less than zero).

Parameters
opContext—Operational context
schedule—— The DatapurgingSchedule instance

Return Value
void
DeviceListApp Service

This chapter describes the DCNM web services’ API methods for the DeviceListApp service.

Information About DeviceListApp Service

The DeviceListApp Device List contains a list of devices managed by DCNM. This feature allows you to add new devices to the list and remove devices from the list. NetworkElementGroup, which is a logical grouping of devices, allows you to create groups, add devices to the group, and assign them to users. A DCNM user can manage only the devices that you, as the administrator, assign to the user. Each user can specify the credentials to log in to devices managed by that user. A user can specify default credentials for a group of devices and also for individual devices. The credentials specified for a device will override the group credentials and the credentials specified for a group will override the default credentials for the user.

addNetworkElements

Adds a list of new devices to the device list. If a device with the specified IP address already exists in the device, this method will throw AppException.

Parameters
- opContext—Operational context
- neCol—List of identifiers corresponding to the devices to be added to the device list.

Return Value
List of AbstractNetworkElement with its AbstractNetworkElementStatus

bindNetworkElementsToGroup

Adds the specified devices from the specified device group

Parameters
- opContext—Operational context
- neGroupInstanceNameId—InstanceNameId of the device group
createNetworkElementGroup

Creates a NetworkElementGroup with the specified name and adds the specified devices to the group.

Parameters
- opContext—Operational context
- neGroupName—Name of the device group
- neInstanceNameIdCol—List of InstanceNameId of devices to be added to the device group

Return Value
NetworkElementGroup

deleteAllNetworkElementGroups

Deletes all the device groups

Parameters
- opContext—Operational context

Return Value
void

deleteAllNetworkElements

Deletes all the devices managed by DCNM. Devices and their associated management information will be deleted.

Parameters
- opContext—Operational context

Return Value
void

deleteNetworkElementGroups

Deletes the specified device groups
### deleteNetworkElements

Deletes the specified devices from the device list. The specified devices will be removed from all the groups. AbstractNetworkElement and its associated information will be deleted from the DB. TODO what do we do the statistics and events related related to the devices?

**Parameters**
- `opContext`—Operational context
- `neGroupInstanceNameIdCol`—List of device group names to be deleted

**Return Value**
- `void`

---

### discover

Discovery API to discover the network based on IP address

**Parameters**
- `opContext`—OpContext
- `seed`—IpAddress of the seed Device
- `cred`—Credentials Used to discover the network
- `hops`—No of hops that are used for discovery
- `incremental`—Boolean to indicate whether it is incremental discovery or not

**Return Value**
- `DiscoveryTaskStatus` for the corresponding discovery task

---

### discoverNetworkElements

Starts a task to discover configuration and status information for the specified devices. At the start of discovery, status of the devices will be set as DISCOVERY. Upon successful discovery, status will be set to MANAGED.

**Parameters**
- `opContext`—Operational context
getAllDiscoveryTasks

API to return status for list of discovery tasks

Parameters

opContext—OpContext

Return Value

List of DiscoveryTaskStatus for all discovery tasks

getAllNetworkElementGroups

Returns all the device groups with the AbstractNetworkElement belonging to the group. AbstractNetworkElementStatus will be initialized for AbstractNetworkElement.

Parameters

opContext—Operational context

Return Value

List of NetworkElementGroup

getAllNetworkElements

Returns all the network elements that are available in the Device List.

Parameters

opContext—Operational context

Return Value

List of AbstractNetworkElement with its AbstractNetworkElementStatus

gGetDiscoveryStatus

Gets discovery status.

Parameters

opContext—TODO

taskIdCol—
getNetworkElementGroups

Returns specified device groups with the AbstractNetworkElement belonging to the group. AbstractNetworkElementStatus will be initialized for AbstractNetworkElement.

Parameters

opContext—Operational context

Return Value

List of NetworkElementGroup

incrementalDiscovery

Performs incremental discovery.

Parameters

opContext—Operational context
neInstanceNameIdCol—List of InstanceNameId corresponding devices to be discovered
userCredentials—userName/password/enableUserName/enablePassword

Return Value

Task ID that identifies the discovery task

modifyNetworkElementGroups

Modifies the device membership for a list of device groups.

Parameters

opContext—Operational context
neGroupCol—List of NetworkElementGroup

Return Value

void

performDiscovery

Discovery API to discover the network based on IP address

Parameters

opContext—Operational context
performIncrementalDiscovery

Parameters

opContext—Operational context

neInstanceNameIdCol—List of InstanceNameId corresponding devices to be discovered

userCredentials—userName/password/enableUserName/enablePassword

Return Value

Task ID that identifies the discovery task

performNetworkElementsDiscovery

Starts a task to discover configuration and status information for the specified devices. At the start of discovery, status of the devices will be set as DISCOVERY. Upon successful discovery, status will be set to MANAGED.

Parameters

opContext—Operational context

neInstanceNameIdCol—List of InstanceNameId corresponding devices to be discovered

Return Value

Task ID that identifies the discovery task

performNetworkElementsRediscovery

Parameters

opContext—Operational context

neInstanceNameIdCol—

userCredentials—
performNetworkElementsUnmanaged

Deletes the information associated with the specified devices. The device status will be marked as UNMANAGED.

Parameters

- `opContext`—Operational context
- `neInstanceNameIdCol`—List of InstanceNameId corresponding to the devices to be unmanaged.

Return Value

void

rediscoverNetworkElements

Rediscovers network elements.

Parameters

- `opContext`—Operational context
- `neInstanceNameIdCol`—
- `userCredentials`—

Return Value

void

unbindNetworkElementsToGroup

Removes the specified devices from the specified device group

Parameters

- `opContext`—Operational context
- `neGroupInstanceNameId`—InstanceNameId of the device group
- `neInstanceNameIdCol`—List of InstanceNameId of devices to be added to the device group

Return Value

void

unmanageNetworkElements

Deletes the information associated with the specified devices. The device status will be marked as UNMANAGED.
unmanageNetworkElements

Send document comments to nexus7k-docfeedback@cisco.com

Parameters
- opContext—Operational context
- neInstanceNameIdCol—List of InstanceNameId corresponding to the devices to be unmanaged.

Return Value
- void
DhcpSnoopingApp Service

This chapter describes the DCNM web services’ API methods for the DhcpSnoopingApp service.

Information About DhcpSnoopingApp Service

This chapter defines the APIs exposed by the DHCP snooping feature service.

addStaticBindings

Creates static binding entries given a list of static binding entry objects.

ValidationException is thrown if any of the following situation occurs:

- If the dhcpSnoopingBindings collection is null or the collection is empty.
- If the dhcpSnoopingBindings collection contains one or more null element, or the collection contains objects that are not of type DhcpSnoopingBinding.
- If any of the DhcpSnoopingBinding, in the dhcpSnoopingBindings collection does not contain value for abstractNetworkElementRef.
- If the AbstractNetworkElement specified by abstractNetworkElementRef in any of the DhcpSnoopingBinding objects in the collection dhcpSnoopingBindings does not exist in the database.
- If any of the DhcpSnoopingBinding, in the dhcpSnoopingBindings collection does not contain value for networkInterfaceRef.
- If the NetworkInterface specified by networkInterfaceRef in any of the DhcpSnoopingBinding objects in the collection dhcpSnoopingBindings does not exist in the database.
- If any of the DhcpSnoopingBinding, in the dhcpSnoopingBindings does not contain value for vlanRef.
- If the (@link com.cisco.dcbu.dcm.model.VLAN.VlanExternal} specified by vlanRef in any of the DhcpSnoopingBinding objects in the collection dhcpSnoopingBindings does not exist in the database.

IntegrityException is thrown if any of the DhcpSnoopingBinding objects in the dhcpSnoopingBindings collection already exist.

PropertiesException is thrown if any of the following situation occurs:
clearRateLimitingConfigurationInInterfaces

Clears the DHCP rate limiting configurations done in a collection of untrusted interfaces. This API also restores the DHCP rate limiting to default value.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or empty or it is not of type InstanceNameId.
- If any of the interface specified by its InstanceNameId in the collection interfaceInstanceNameIds does not exist in the database.

**Parameters**

opContext—Operational context

interfaceInstanceNameIds— A collection of InstanceNameId of interfaces.

**Return Value**

void

deleteAllBindings

Deletes both static and dynamic entries configured in all the VLANs in a given interface, given the InstanceNameId of the interfaces.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or not a valid interface instance name ID.
- If the interface does not exist in the database.
- If the interface does not contain any DhcpSnoopingBinding associated to it.

**Parameters**

opContext—Operational context

interfaceInstanceNameIds— A collection of InstanceNameId of the interfaces whose bindings are to be deleted.
deleteAllBindingsInNetworkElements

Deletes both static and dynamic entries configured in the network elements, given the InstanceNameId of the network elements.

ValidationException is thrown if any of the following situation occurs:
- If the argument passed is null or not a valid network element instance name ID.
- If the network element does not exist in the database.
- If the network element does not contain any DhcpSnoopingBinding associated to it.

Parameters
opContext—Operational context
networkElementIds— A collection of InstanceNameId of the network elements in which the bindings are to be deleted.

Return Value
void

deleteAllDynamicBindingsInInterfaces

Deletes all dynamically learned entries corresponding to a collection of interfaces in binding table, given the InstanceNameId of the interfaces.

ValidationException is thrown if any of the following situation occurs:
- If the argument passed is null or not a valid interface instance name ID.
- If the interface does not exist in the database.
- If the interface does not contain any DhcpSnoopingBinding associated to it

Parameters
opContext—Operational context
interfaceInstanceNameIds— A collection of InstanceNameId of the interfaces in which bindings are to be deleted.

Return Value
void

deleteAllDynamicBindingsInNetworkElements

Deletes all dynamic binding entries in a given collection of network elements, given the InstanceNameId of the network elements.

ValidationException is thrown if any of the following situation occurs:
deleteAllDynamicBindingsInVlanOfAnInterface

If the argument passed is null or not a valid network element instance name ID.
If the network element does not exist in the database.
If the network element does not contain any DhcpSnoopingBinding associated to it

Parameters
opContext—Operational context
networkElementIds—A collection of InstanceNameId of the network element in which bindings are to be deleted.

Return Value
void

deleteAllDynamicBindingsInVlanOfAnInterface

Deletes all dynamic binding entries configured in a given VLAN in an interface, given the InstanceNameId of the interface and VLAN.
ValidationException is thrown if any of the following situation occurs:
• If the interfaceInstanceNameId argument passed is null or not a valid interface instance name ID.
• If the interface does not exist in the database.
• If the vlanInstanceNameId argument passed is null or not a valid VLAN instance name ID.
• If the VLAN does not exist in the database.
• If the VLAN does not contain any DhcpSnoopingBinding associated to it

Parameters
opContext—Operational context
vlanInstanceNameId—InstanceNameId of the VLAN in which the bindings are to be deleted.
interfaceInstanceNameIds—A collection of InstanceNameId of the interfaces in which bindings are to be deleted for the given VLAN.

Return Value
void

deleteAllStaticBindingsInInterfaces

Deletes all static binding entries configured in a given collection of interfaces, given the InstanceNameId of the interfaces.
ValidationException is thrown if any of the following situation occurs:
• If the argument passed is null or not a valid interface instance name ID.
• If the interface does not exist in the database.
• If the interface does not contain any DhcpSnoopingBinding associated to it
deleteAllStaticBindingsInNetworkElements

Deletes all static binding entries in a given collection of network elements, given the InstanceNameId of the network elements.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or not a valid network element instance name ID.
- If the network element does not exist in the database.
- If the network element does not contain any DhcpSnoopingBinding associated to it.

Parameters

- opContext—Operational context
- networkElementIds—A collection of InstanceNameId of the network elements in which bindings are to be deleted.

Return Value

void

deleteAllStaticBindingsInVlanOfAnInterface

Deletes all static binding entries configured in a given VLAN in an interface, given the InstanceNameId of the interface and VLAN.

ValidationException is thrown if any of the following situation occurs:

- If the argument interfaceInstanceNameId is null or not a valid interface instance name ID.
- If the interface does not exist in the database.
- If the argument vlanInstanceNameId is null or not a valid VLAN instance name ID.
- If the VLAN does not exist in the database.
- If the VLAN does not contain any DhcpSnoopingBinding associated to it.

Parameters

- opContext—Operational context
- interfaceInstanceNameId—InstanceNameId of the interface in which bindings are to be deleted for the given VLAN.
- vlanInstanceNameId—InstanceNameId of the VLAN in which the bindings are to be deleted.

Return Value

void
**deleteDynamicBindings**

Deletes dynamically learned entries in binding table, given the InstanceNameId of the DHCP snooping binding.

ValidationException is thrown if any of the following situation occurs:

- If the collection dhcpSnoopingBindings is null or empty.
- If any of the element in the collection dhcpSnoopingBindings is null or not a valid DHCP snooping binding instance name ID.
- If the DHCP snooping binding does not exist in the database.

**Parameters**

- opContext—Operational context
- dhcpSnoopingBindings—A collection of InstanceNameId of the DHCP snooping bindings that are to be deleted.

**Return Value**

void

---

**deleteStaticAndDynamicBindings**

Deletes both static and dynamic entries configured in a particular VLAN in an interface, given the InstanceNameId of the interface and VLAN.

ValidationException is thrown if any of the following situation occurs:

- If the interfaceInstanceNameId argument passed is null or not a valid interface instance name ID.
- If the interface does not exist in the database.
- If the vlanInstanceNameId argument passed is null or not a valid VLAN instance name ID.
- If the VLAN does not exist in the database.
- If the VLAN does not contain any DhcpsnoopingBinding associated to it.

**Parameters**

- opContext—Operational context
- vlanInstanceNameId—InstanceNameId of the VLAN in which the bindings are to be deleted.
- interfaceInstanceNameIds—A collection of InstanceNameId of the interfaces in which bindings are to be deleted for the given VLAN.

**Return Value**

void
**deleteStaticBindings**

Deletes static binding entries from the DHCP snooping binding table, given the InstanceNameId of the DHCP snooping bindings.

ValidationException is thrown if any of the following situation occurs:

- If the collection dhcpSnoopingBindings is null or empty.
- If any of the element in the collection dhcpSnoopingBindings is null or not a valid DHCP snooping binding instance name ID.
- If the DHCP snooping binding does not exist in the database.

**Parameters**
- opContext—Operational context
- dhcpSnoopingBindings—A collection of InstanceNameId of the DHCP snooping bindings that are to be deleted.

**Return Value**
- void

**disableDhcpSnoopingInNetworkElement**

Disables DHCP snooping in a network element, given its instance name ID.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

**Parameters**
- opContext—Operational context
- neInstanceNameId—InstanceNameId of network element in which DHCP snooping is to be disabled.

**Return Value**
- void

**disableDhcpSnoopingOnVlans**

Disables DHCP snooping in VLANs, given the list of InstanceNameId of the VLANs.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid VLAN InstanceNameId.
- If the VLAN does not exist in the database.

**Parameters**
- opContext—Operational context
disableDhcpSnoopingService

Disables DHCP snooping Service in a InstanceNameId network element. Service Enabling/Disabling is supported in DC OS platform. If this API is called with the network elements of Catalyst 6500 series switches then FeatureException will be thrown.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameIdCol is null.
- If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters
  opContext—Operational context
  neInstanceNameIdCol—— A collection of InstanceNameId of the network elements.

Return Value
  void

enableDhcpSnoopingOnVlans

Enables DHCP snooping in VLANs, given the list of InstanceNameId of the VLANs.

ValidationException is thrown if any of the following situation occurs:
- If the argument passed is null or it is not a valid VLAN InstanceNameId.
- If the VLAN does not exist in the database.

Parameters
  opContext—Operational context
  vlanInstanceNameIds—- List of InstanceNameId of VlanExternal in which DHCP snooping has to be enabled.

Return Value
  void

enableDhcpSnoopingOnVlansByRange

Enables DHCP snooping in a pre-provisioned VLAN. It is possible to enable DHCP Snooping just by using VLAN IDs. The VLANs in which DHCP snooping has to be enabled need not actually exist in the device. This API addresses this pre-provisioning configuration.
enableDhcpSnoopingService

Enables DHCP snooping Service in a InstanceNameId network element. Service Enabling/Disabling is supported in DC OS platform. If this API is called with the network elements of Catalyst 6500 series switches then FeatureException will be thrown.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null.
- If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters

- opContext—Operational context
- neInstanceNameIdCol— A collection of InstanceNameId of the network elements.

Return Value

void

defendDhcpSnoopingDisabledVlansInNetworkElement

Returns a collection of DHCP snooping disabled VLANs in a given network element given the InstanceNameId of the network element.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters

- opContext—Operational context
- neInstanceNameId— InstanceNameId of the network element.
Chapter 10      DhcpSnoopingApp Service

getDhcpSnoopingEnabledVlansInNetworkElement

Returns a collection of DHCP snooping enabled VLANs in a given network element given the
InstanceNameId of the network element.
ValidationException is thrown if any of the following situation occurs:
• If the argument passed is null or it is not a valid network element InstanceNameId.
• If the network element does not exist in the database.

Parameters
opContext—Operational context
networkElementId—InstanceNameId of the network element.

Return Value
A list of DHCP snooping enabled VlanExternal objects. In the returned list, only the association with
DhcpSnoopingSetting will be present and other associations will be cleared.

getDhcpSnoopingGlobalSettingOnNetworkElements

Returns a collection of DHCP snooping global settings, given the InstanceNameId of the network
elements.
ValidationException is thrown if any of the following situation occurs:
• If the argument passed is null or empty or if any of the element in the collection is not a valid
network element InstanceNameId.
• If the network element does not exist in the database.

Parameters
opContext—Operational context
eInstanceNameIds—A collection of InstanceNameId of the network elements.

Return Value
A collection of DhcpSnoopingGlobalSetting objects for network elements.

getDhcpSnoopingServiceStateInNetworkElements

Returns state of DhcpSnoopingService like whether DhcpSnoopingService is enabled or disabled in a
list of network elements. Given the list of instance name IDs of the network elements, returns a array of
Boolean values.
ValidationException is thrown if any of the following situation occurs:
getDhcpSnoopingSettingOnVlans

Returns the DHCP snooping settings associated with the VLANs, given the list of InstanceNameId of VLANs

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid VLAN InstanceNameId.
- If the VLAN does not exist in the device.

Parameters

- opContext—Operational context
- vlanInstanceNameIds—A list of InstanceNameId of VLANs to be queried for DHCP snooping setting.

Return Value

A list of DhcpsnoopingSetting objects. The returned list will contain the DhcpsnoopingSetting objects pertaining to the given list of VLANs

getDynamicBindingsInInterface

Returns the dynamic bindings that are learnt in an interface, given the InstanceNameId of the interface.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.

Parameters

- opContext—Operational context
- interfaceInstanceNameId—InstanceNameId of the interface.
getInterfacesWithDhcpRateLimitingInNetworkElement

Returns all the interfaces having DHCP rate configured, given the InstanceNameId of network element. ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the network element.

Return Value

A collection of interface objects having DHCP rate limiting configured. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- TrustStateSetting association.
- NetworkInterfaceName association.

getInterfacesWithDynamicBindingsInNetworkElement

Returns the interfaces having dynamic bindings in them, given the InstanceNameId of the network element. The network element is the device in which the interface is present.

ValidationException is thrown any of the following situations occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of network element

Return Value

A collection of interfaces in the network element in which dynamic bindings are learned. Returned collection will have InstanceNameIds of interface objects having dynamic bindings. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- DhcpSnoopingBinding association.
getInterfacesWithStaticBindingsInNetworkElement

Returns the interfaces having static bindings configured in them, given the InstanceNameId of the network element. The network element is the device in which the interface is present. ValidationException is thrown any of the following situations occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

**Parameters**

- **opContext**—Operational context
- **neInstanceNameId**—InstanceNameId of network element

**Return Value**

A collection of interfaces in the network element in which static bindings are configured. Returned collection will have InstanceNameIds of interface objects having static bindings. In the returned list of objects, only the following associations will be present, and all other associations will be cleared:

- DhcpSnoopingBinding association.
- NetworkInterfaceName association.

getNumberOfDynamicBindingsInVlans

Returns the number of dynamic bindings configured in the VLANs given the InstanceNameId of the VLANs. ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the VLAN does not exist in the device.

**Parameters**

- **opContext**—Operational context
- **vlanInstanceNameIds**—A collection of InstanceNameId of VLANs.

**Return Value**

A collection of integer values representing the number of dynamic bindings configured in VLANs.

getNumberOfStaticBindingsInVlans

Returns the number of static bindings configured in a collection of VLANs, given the InstanceNameId of the VLANs. ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid VLAN InstanceNameId.
getStaticBindingsInInterface

Returns the static bindings configured in an interface, given the InstanceNameId of the interface.
ValidationException is thrown if any of the following situation occurs:
- If the argument passed is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.

Parameters
- opContext—Operational context
- interfaceInstanceNameId—- InstanceNameId of the interface

Return Value
A collection of static binding configured in the given interface. Returned collection will have objects of type DhcpSnoopingBinding. In the returned list of objects, only the following associations will be present and other associations will be cleared.
- AbstractNetworkElement association.
- NetworkInterface association.
- VlanExternal association.

getStaticBindingsInVlanOfAnInterface

Returns the static bindings configured in a VLAN in an interface, given the InstanceNameId of the VLAN and the interface.
ValidationException is thrown if any of the following situation occurs:
- If the argument interfaceInstanceNameId passed is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.
- If the argument vlanInstanceNameId passed is null or it is not a valid VLAN InstanceNameId.
- If the VLAN does not exist in the database.

Parameters
- opContext—Operational context
- interfaceInstanceNameId—- InstanceNameId of the interface
- vlanInstanceNameId—- InstanceNameId of a VLAN in the interface
Return Value
A collection of static binding configured in given VLAN in an interface. Returned collection will have objects of type DhcpSnoopingBinding. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- AbstractNetworkElement association.
- NetworkInterface association.
- VlanExternal association.

getStaticDhcpSnoopingBindings

Returns a collection of DHCP snooping binding objects corresponding to the InstanceNameIds given.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid DHCP Snooping Binding InstanceNameId.
- If the DHCP Snooping Binding does not exist in the database.

Parameters

- opContext—Operational context
- dhcpSnoopingBindings—a collection of InstanceNameId of DHCP snooping binding objects representing entries in the DHCP snooping binding table.

Return Value
A collection of DhcPSnoopingBinding objects DhcpSnoopingBinding. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- AbstractNetworkElement association.
- NetworkInterface association.
- VlanExternal association.

getTrustStateSettingOnInterfaces

Returns the DHCP snooping trust state settings configured in a collection of Layer 2 interfaces, given the InstanceNameId of the interfaces.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.

Parameters

- opContext—Operational context
- interfaceInstanceNameIds— List of InstanceNameId of interfaces.

Return Value
A collection of TrustStateSetting objects corresponding to the given interfaces TrustStateSetting.
getUntrustedInterfacesWithDefaultRateLimitInNetwork Element

Returns all the untrusted interfaces having default DHCP rate, in a network element, given the InstanceNameId of network element.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId of the network element.

Return Value
A collection of untrusted interface objects having default DHCP rate. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- TrustStateSetting association.
- NetworkInterfaceName association.

getVlansWithDhcpSnoopingInNetworkElement

Returns a collection of VLANs which has DHCP snooping setting (DHCP snooping enabled and disabled) in a given network element given the InstanceNameId of the network element. This will not return the VLANs in which DHCP snooping is not supported.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId of the network element.

Return Value
A list of DHCP snooping enabled and disabled VlanExternal objects. In the returned list, only the associations with DhcpSnoopingSetting will be present and other associations will be cleared.

modifyDhcpSnoopingGlobalSetting

Modifies one or more attributes of DhcpSnoopingGlobalSettings. Modification of "dhcpSnoopingServiceEnable" is not allowed in this API. If it is modified then MetadataException will be thrown. To enable/disable DHCP Snooping Service API.

ValidationException is thrown if any of the following situation occurs:
modifyStaticBindings

Modifies one or more DHCP snooping binding objects that exist in the database.

ValidationException is thrown if any of the following situation occurs:
- If the collection dhcpSnoopingBindings is null or empty.
- If any of the element in the collection dhcpSnoopingBindings is null.
- If any of the element in the collection dhcpSnoopingBindings is not of type DhcpSnoopingBinding.

PropertiesException is thrown if any of the attributes of DhcpSnoopingBinding is not valid.

Example:
- If the value for IpAddress is not specified.
- If the value for the MacAddress is not specified
- If the value for the lease expiry time is not specified etc.

Parameters
opContext—Operational context
dhcpSnoopingBindings—A collection of modified DhcpSnoopingBinding entries.

Return Value
void

modifyTrustStateSettings

Modifies one or more existing trust state settings in the given collection of interfaces.

ValidationException is thrown if any of the following situation occurs:
- If interfaceInstanceNameIds collection is null or it is empty.
- If interfaceInstanceNameIds collection contains an object that is not of type NetworkInterface.
modifyTrustStateSettings

- If TrustStateSetting is not supported in that interface. It is supported only in the interfaces of type SwitchedNetworkInterface and RoutedNetworkInterface.
- If trustStateSettings collection is null or it is empty.
- If trustStateSettings collection contains an object that is not of type TrustStateSetting.

IntegrityException is thrown if the interfaceInstanceNameIds and trustStateSettings collections size are not equal.

PropertiesException is thrown, if any of the attribute of TrustStateSetting is not valid.

Example: TrustState attribute in the TrustStateSetting object cannot have Trusted DHCP trust state, if the interface in which the TrustStateSetting is configured is a RoutedNetworkInterface.

Parameters

opContext—Operational context
interfaceInstanceNameIds—- A collection of InstanceNameIds of interfaces.
trustStateSetting—- A collection of modified TrustStateSetting object TrustStateSetting

Return Value

void
This chapter describes the DCNM web services’ API methods for the Dot1xApp service.

**Information About Dot1xApp Service**

IEEE 802.1X defines the framework for port-based network access control. 802.1X uses the physical characteristics of the device to authenticate and authorize devices attached to a switch port and prevents access to that port in cases when authentication and authorization fails. The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Modify APIs—Modify existing Dot1x interface and global settings.
- Enable and Disable APIs—Enable and disable Dot1x in the device.

**disableDot1x**

Disables dot1x authentication on one or more network elements. This API is to disable system-auth-control option globally in the device.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If dot1xNetworkInterfaceInstanceNameIdCol is not a valid InstanceNameId of AbstractNetworkElement objects.

**Parameters**

- opContext—Operational context
- neInstanceNameIdCol—InstanceNameId of one or more dot1x enabled network elements.

**Return Value**

- void
disableDot1xInInterfaces

Disables the dot1x settings for one or more interfaces. Given the InstanceNameId of one or more dot1x enabled interfaces, it disables dot1x in the corresponding interfaces.

ValidationException is thrown if any of the following situation occurs:

- If dot1xNetworkInterfaceInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If dot1xNetworkInterfaceInstanceNameIdCol is not a valid InstanceNameId of Dot1xNetworkInterfaceSetting object.

Parameters
- opContext—Operational context
- dot1xNetworkInterfaceInstanceNameIdCol—InstanceNameId if one or more NetworkInterface objects.

Return Value
- void

disableDot1xService

Disables dot1x authentication on one or more network elements. This API is to disable dot1x service globally in the device.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If neInstanceNameIdCol does not a valid AbstractNetworkElement InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameIdCol—InstanceNameId of one or more dot1x enabled network elements.

Return Value
- void

enableDot1x

Enables dot1x authentication on one or more network elements. Given the instance name ID of one or more network elements, dot1x will be enabled on those elements. This API is to enable system-auth-control option globally in the device.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If dot1xNetworkInterfaceInstanceNameIdCol is not a valid InstanceNameId of AbstractNetworkElement.

IntegrityException is thrown if any of the following situation occurs:
enableDot1xInInterfaces

If any one of the NetworkInterface has one of the following feature enabled as well as dot1x configured in them:

- NetworkInterface is a span destination interface.
- NetworkInterface is a Port Security Enabled interface.
- NetworkInterface is a Voice VLAN Enabled Interface.

**Parameters**

- opContext—Operational context
- neInstanceNameIdCol—List of dot1x disabled network elements instance name ID.

**Return Value**

void

enableDot1xInInterfaces

Enables the dot1x settings for one or more interfaces. Given the InstanceNameId of one or more dot1x enabled interfaces, returns the corresponding dot1x interface settings objects.

ValidationException is thrown if any of the following situation occurs:

- If dot1xNetworkInterfaceInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If dot1xNetworkInterfaceInstanceNameIdCol is not a valid InstanceNameId of Dot1xNetworkInterfaceSetting object.

**Parameters**

- opContext—Operational context
- dot1xNetworkInterfaceInstanceNameIdCol—InstanceNameId if one or more NetworkInterface objects.

**Return Value**

A collection of Dot1xNetworkInterfaceSetting objects whose port control set as force-authorized. The returned objects will have their corresponding NetworkInterface reference. But If that NetworkInterface has any other associations, those will be cleared.

enableDot1xService

Enables dot1x authentication on one or more network elements. Given the instance name ID of one or more network elements, dot1x will be enabled on those elements. This API is to enable dot1x service option globally in the device.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If neInstanceNameIdCol does not a valid AbstractNetworkElement InstanceNameId.

**Parameters**

- opContext—Operational context
- neInstanceNameIdCol—List of InstanceNameId of dot1x disabled devices.
getDot1xConflictingPorts

If the platform type is a Catalyst 6500 series switch, returns the list of network interface objects that has been configured with dot1x whose port control is not Force_Authorized as well as one of the following:

- NetworkInterface is a span destination interface.
- NetworkInterface is a Port Security Enabled interface.
- NetworkInterface is a Voice VLAN Enabled Interface.
- NetworkInterface is of type TRUNK

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null or it is not of type InstanceNameId.

If the platform type is Nexus 7000 series switch, returns an empty collection.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of AbstractNetworkElement objects.

Return Value

A collection of NetworkInterface objects.

g DOT xConflictingPorts

getDot1xGlobalSetting

Returns the device level dot1x settings for one or more network elements. Given the InstanceNameId of one or more network elements, returns the corresponding Dot1xGlobalSetting object for those network elements.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If neInstanceNameIdCol is not a valid InstanceNameId of AbstractNetworkElement object.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—InstanceNameId of one or more AbstractNetworkElement objects.

Return Value

A collection of Dot1xGlobalSetting objects.

g DOT xInterfacesSetting

Returns the dot1x settings for one or more interfaces. Given the InstanceNameId of one or more dot1x enabled interfaces, returns the corresponding dot1x interface settings objects.
getDot1xNetworkInterfacesInNetworkElement

Returns the dot1x interface setting objects in a network element having the specified port control state. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null or it is not of type InstanceNameId.
- If neInstanceNameId is not a valid network element InstanceNameId.

In PortControl state collection, user can send one or more of the following dot1x port control states:

- Auto
- Force_Authorized
- Force_Unauthorized
- Disabled

For example:

- If user passes port control as Auto, and Force_Authorized, then: This API will return all the dot1x interface setting objects for which the PortControl attribute value is Auto or Force_Authorized.
- If the PortControl state collection is NULL, then it will return all the dot1x interfaces setting objects, irrespective of their PortControl state.

Note: Disable means, the interfaces in which Dot1x can not be enabled. Following are the interfaces in which dot1x can not be enabled:

- All physical interfaces that are of type TRUNK
- All physical interfaces that are of type Private_VLAN
- All physical interfaces that are configured as SPAN destinations
- All logical interfaces

Parameters

opContext—Operational context
neInstanceNameId—InstanceNameId of the network element.
dot1xPortControlStateFilter—one or more Dot1xPortControl state enumerations.
**modifyDot1xGlobalSetting**

Method to modify device level dot1x settings.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameId is null or it is not a valid InstanceNameId of a network element.
- If Dot1xGlobalSetting is null or the object does not exist in the database.

IntegrityException is thrown if any of the following situation occurs:
- If any one of the NetworkInterface has one of the following feature enabled as well as dot1x configured in them:
  - NetworkInterface is a span destination interface.
  - NetworkInterface is a Port Security Enabled interface.
  - NetworkInterface is a Voice VLAN Enabled Interface.

**Parameters**

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the network element.
- dot1xGlbSetting—modified (@link Dot1xGlobalSetting) object.

**Return Value**

void

**modifyDot1xNetworkInterfacesSetting**

Modifies the dot1x settings for one or more interfaces.

ValidationException is thrown if any of the following situation occurs:
- If dot1xNetworkInterfaceSettingCol is null or empty or it is not of type Dot1xNetworkInterfaceSetting.
- Only existing dot1x interfaces setting (exist in database) can be modified.

PropertiesException is thrown if any of the following situation occurs:
- In the dot1xNetworkInterfaceSettingCol collection, if any one of the Dot1xNetworkInterfaceSetting attribute is not valid.

Example:
- quietPeriod of a Dot1xNetworkInterfaceSetting is out of range.

IntegrityException is thrown if any of the following situation occurs:
- If the dot1xNetworkInterfaceSettingCol collection contains a Dot1xNetworkInterfaceSetting object that is associated with a NetworkInterface which comes with one of the following:
Chapter 11      Dot1xApp Service

modifyDot1xNetworkInterfacesSetting

- NetworkInterface is a logical interface.
- NetworkInterface is a span destination interface.
- NetworkInterface is a Port Security Enabled interface.
- NetworkInterface is a Voice VLAN Enabled Interface.

**Parameters**

opContext—Operational context
dot1xNetworkInterfaceSettingCol—a collection of modified (@link Dot1xNetworkInterfaceSetting) objects.

**Return Value**

void
EtherChannelApp Service

This chapter describes the DCNM web services’ API methods for the EtherChannelApp service.

Information About EtherChannelApp Service

The port-channel technology allows you to scale the link bandwidth by aggregating, or bundling, parallel links. Two to eight links of Fast Ethernet (FE) or Gigabit Ethernet (GE) ports can be bundled into one logical link. Port channels also provide redundancy. If one link in the port channel goes down, traffic still flows through the port channel via another link that is up in the port channel. This situation does not bring down the port-channel interface, and a recomputation of the route in the network is not required. A higher aggregated bandwidth is obtained by the sum of the bandwidth of all the ports in the channel. The data traffic is load balanced across the member links, based on the hash algorithm.

The port-channel interfaces are logical interfaces (ports) that have a unique port channel ID within a device. A port-channel interface can be either a Layer 2 port-channel interface (SwitchedEtherChannelNetworkInterface) or a Layer 3 port-channel interface (RoutedEtherChannelNetworkInterface). Each port-channel interface can have 0 to 16 physical ports as members for Catalyst 6500 series switches, but those member ports should have common operational and configurational parameters. A Layer 2 port channel can be associated with Layer 2 physical ports only and a Layer 3 port channel can be associated with Layer 3 physical ports only.

The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Create APIs—Create new port channels.
- Modify APIs—Modify basic attributes of existing port-channel interfaces.
- Delete APIs—Delete existing port channels.
- Add and Remove APIs—Add and remove the association between the port channel and its member ports.
- Enable and Disable APIs—Enable and disable LACP service in the device.

**addPortsToEtherChannelEndPoint**

Associates the given ports to the given Ethernet channel end point. The given ports should operate in the same mode (switched or routed) as the given Ethernet channel end point.

Member port mode will be updated based on Ethernet channel end point protocol as follows:
createEtherChannel

ProtocolDefault Mode
LACPActive
NONEOn
ValidationException is thrown if any of the following situation occurs:
- If channelEpId is null.
- If channelEpId is not a valid Ethernet channel end point InstanceNameId.
- If the portIds collection is null or the collection is empty or the collection contains objects other than InstanceNameId of physical ports.
- If port security is enabled in any one of the given member ports (802.1x ports).
- If member ports are configured with different ACLs.
- If any one of the member ports is a Switched Port Analyzer (SPAN) destination
PropertiesException is thrown if any of the following situation occurs:
- If given Ethernet channel end point has 16 member port already.
- If given ports operate in different mode (ex : switched) than the given Ethernet channel end point(ex : routed).
IntegrityException is thrown if any of the following situation occurs:
- If the given channel end point does not exist in device.
- If the given ports do not exist in device.

Parameters
opContext—Operational context
channelEpId—InstanceNameId of the SwitchedEtherChannelNetworkInterface/RoutedEtherChannelNetworkInterface to which the given ports should be associated
portIds—list of InstanceNameId of ports (SwitchedNetworkInterface / RoutedNetworkInterface) to be added

Return Value
void

createEtherChannel

Creates the given new Ethernet channel and returns the list of instance name ids of newly created Ethernet channel endpoints.
The passed Ethernet channel object should have its both Ethernet channel network interface endpoints and their corresponding associations with channel members populated.
ValidationException is thrown if any of the following situation occurs:
- If newChannel is null.
- If Protocol is not set to NONE for any of the Ethernet channel end point which does not have member ports.
- If Protocol is set to PAgP for any of the Ethernet channel end point.
createEtherChannelForSpecifiedEndpoints

Creates an Ethernet channel between the given ports with the given channel id.

This method creates endpoints using the given channel id and the given list of ports. The mode (switched / routed) of new Ethernet channel created will be based on the mode of given ports. All the given ports should be of same mode either as switched or routed. All the attributes of the Ethernet channel network interface endpoints and their association parameters are set to default values. If the given member ports list is empty, the protocol is set to NONE else the protocol and mode are set to default values: LACP and ACTIVE.

ValidationException is thrown if any of the following situation occurs:

- If sourceEpChannelId is null.
- If the sourceEpPortMemberIds is not a valid Network Interface InstanceNameId.
- If the neighborEpChannelIds contains one or more null element, or empty

PropertiesException is thrown if any of the following situation occurs:

- If sourceEpChannelId is not valid Ethernet channel ID.
- If neighborEpChannelIds does not contain valid Ethernet channel IDs.

Example:

- Ethernet channel ID should be in the range between 1 to 256 in Catalyst 6500 series switches, and 1 to 4096 in Nexus 7000 series switch.

IntegrityException is thrown if any of the following situation occurs:

Example:

- Ethernet channel ID should be in the range between 1 to 256 in Catalyst 6500 series switches, and 1 to 4096 in Nexus 7000 series switch.
**deleteEtherChannelEndPoints**

Deletes all the given Ethernet channel endpoints.

This method deletes all the given instances of SwitchedEtherChannelNetworkInterface / RoutedEtherChannelNetworkInterface endpoints and their port member associations.

ValidationException is thrown if any of the following situation occurs:

- If endPointIds collection is null or it is empty.
- If endPointIds collection contains an element that is not of type SwitchedEtherChannelNetworkInterface / RoutedEtherChannelNetworkInterface InstanceNameId.

IntegrityException is thrown if the given port channel end points do not exist in device.

**Parameters**

- opContext—Operational context
- endPointIds—array of InstanceNameId of endpoints (SwitchedEtherChannelNetworkInterface / RoutedEtherChannelNetworkInterface)

**Return Value**

void

**deleteEtherChannels**

Deletes all the given ether channels.

This method deletes each given Ethernet channel by removing all the port members and the Ethernet channel endpoints of the given etherchannels.

ValidationException is thrown if any of the following situation occurs:

- If the Ethernet channel network interface with the given Ethernet channel ID already exist in the database.
- If the sourceEpPortMemberIds collection contains a NetworkInterface InstanceNameId that does not exist in the database.
- If a NetworkInterface in the sourceEpPortMemberIds contains duplicate NetworkInterface objects.

**Parameters**

- opContext—Operational context
- sourceEpChannelId—channel id to be used for the source endpoint
- sourceEpPortMemberIds—List of InstanceNameId of source endpoint port members (SwitchedNetworkInterface / RoutedNetworkInterface)
- neighborEpChannelIds—List of channel ids (Integer) to be used for the neighbor endpoints
- neighborEpPortMemberIds—List of List objects containing InstanceNameId of neighbor endpoint port members (SwitchedNetworkInterface/ RoutedNetworkInterface)

**Return Value**

The List of InstanceNameId of the new channel endpoints

Send document comments to nexus7k-docfeedback@cisco.com
disableLacp

Disables LACP service on one or more network element. This API is applicable only for devices running Cisco NX-OS.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameIdCol is null.
- If the neInstanceNameIdCol contains one or more null element, or the collection is empty or it is not type InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—list of InstanceNameId of the Network Element for which the LACP should be disabled

Return Value

- void

enableLacp

Enables LACP service on one or more network element. This API is applicable only for devices running Cisco NX-OS.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameIdCol is null.
- If the neInstanceNameIdCol contains one or more null element, or the collection is empty or it is not type InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—list of InstanceNameId of the Network Element for which the LACP should be enabled

Return Value

- void
getAllEtherChannels

Returns all the etherchannels present in the network.

The etherchannels returned by this method will have its both source and neighbor endpoints populated. Please note that Ethernet channel network interface endpoints will have all the attributes populated except the port member associations. These associations can be obtained using other specific get API, getEtherChannelLinks(OpContext, List).

Please note that the returned List will contain only one Ethernet channel instance per Ethernet channel in network level. This instance will have any one of the port-channel endpoints as source endpoint and other port-channel endpoints as neighbor endpoints.

Parameters
opContext—Operational context

Return Value
All the etherchannels in network. The returned list will contain the list of either Ethernet channel instances.

Following associations will be there for an Ethernet channel:
- Source end of an Ethernet channel
- Neighbor end of an Ethernet channel

Following associations will be there for each end of an Ethernet channel:
(other associations will be cleared)
- Interface status
- Interface Capability
- Interface Setting

getEtherChannelGlobalSettings

Returns the Ethernet channel global setting of the given network elements.

The Ethernet channel global setting objects will be ordered in the returned List based on the order of given network element instance ids.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameIds.

Parameters
opContext—Operational context

networkElementIds—InstanceNameId of the network element instances whose Ethernet channel global setting information is required
Chapter 12      EtherChannelApp Service

Send document comments to nexus7k-docfeedback@cisco.com

getEtherChannelLinks

Returns all the Ethernet channel links present in the given Ethernet channel endpoints. Each link will have a collection (both ends) of physical interfaces. If the link does not exist, the collection will have only one end in the list.

Following associations will be available for each end of the link:
SwitchedNetworkInterface/RoutedNetworkInterface:
  • SwitchedEtherChannelPortSetting/RoutedEtherChannelPortSetting
  • SwitchedEtherChannelPortStatus/RoutedEtherChannelPortStatus

ValidationException is thrown if the argument passed is null or it is not a valid Switched/Routed Ethernet channel interface InstanceNameIds.

Parameters
opContext—Operational context
channelEndPointIds—InstanceNameId of the SwitchedEtherChannelNetworkInterface and RoutedEtherChannelNetworkInterface instances whose links information is required

Return Value
The Ethernet channel links present in the given etherchannels. The returned List will contain the List objects containing either NetworkInterfaceLink instances or NetworkInterfaceLink instances.

getEtherChannelsInNetworkElement

Returns all the etherchannels that have atleast one end point in the given network element. The etherchannels returned by this method will have its both source and neighbor endpoints populated. Please note that Ethernet channel network interface endpoints will have all the attributes populated except the port member associations. These associations can be obtained using other specific API get method, getEtherChannelLinks(OpContext, List).

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
networkElementId—InstanceNameId of the network element for which the etherchannels are required

Return Value
The etherchannels present in the given network element. The returned list will contain the list of either Ethernet channel instances.

Following associations will be there for an Ethernet channel:
  • Source end of an Ethernet channel
getLacpStateOfNetworkElements

Returns state of LACP service whether LACP is enabled or disabled in a list of network elements. Given the list of instance name IDs of the network elements, returns the list of Boolean values.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId
- if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

opContext—Operational context
neInstanceNameIdCol—InstanceNameId of the one or more Network Element for which the AAA state is required

Return Value

The returned list will contain Boolean instances.

Boolean value TRUE represents LACP is enabled in the given network element.

Boolean value FALSE represents LACP is disabled in the given network element.

getNetworkCardsWithLoadBalanceSettings

Returns all the network cards of the given network element with their Ethernet channel load balance setting configurations populated.

The returned List will contain instances of NetworkCard.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

opContext—Operational context
neInstanceNameId—InstanceNameId of the network element for which module level load balance settings are required

Return Value

List of NetworkCard instances. In the returned list of objects, only Load balance setting associations will be present,
modifyEtherChannelGlobalSetting

Modifies the Ethernet channel global setting of the given network element with the given global setting object.

ValidationException is thrown if any of the following situation occurs:
- if the argument passed is null or it is not a valid network element InstanceNameId.
- If networkCardSlotNos collection contains an element that is null or not Integer type
- If settings collection contains an element that is null or is not type NetworkCardEtherChannelLoadBalanceSetting

Parameters
- opContext—Operational context
- networkElementId—InstanceNameId of the network element whose global setting has to be updated
- gSetting—new global setting object to be set

Return Value
- void

modifyEtherChannelInterfaces

Modifies basic attributes (for example, Speed, duplex, etc) of an existing Ethernet channel end point with the modifications in the given Ethernet channel end point list.

This method updates the server with modifications in basic attributes of Ethernet channel endpoint only. Modification of port member association changes are not done in the given Ethernet channel end point.

ValidationException is thrown if any of the following situation occurs:
- If the portChannelNetworkInterfaces is null or empty.
- If the portChannelNetworkInterfaces contains one or more null element, or the collection contains objects that are not of type SwitchedEtherChannelNetworkInterface or RoutedEtherChannelNetworkInterface.

IntegrityException is thrown if any of the following situation occurs:
- If the given port channel network interfaces do not exist in device.

Use addPortsToEtherChannelEndPoint(opContext, InstanceNameId, List)/#removePortsFromEtherChannelEndPoint(InstanceNameId, List}) for modifying port member associations.

Use modifyNetworkInterfaceLinks(opContext, List)—For modifying all the attributes in the port members

Parameters
- opContext—Operational context
- portChannelNetworkInterfaces—List of Modified Ethernet channel network interfaces (SwitchedEtherChannelNetworkInterface/RoutedEtherChannelNetworkInterface)
modifyNetworkCardEtherChannelLoadBalanceSettings

Modifies the Ethernet channel load balance setting configuration of the given network cards in the given network element.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If networkCardSlotNos collection contains an element that is null or empty.
- If settings collection contains an element that is null or empty.
- If the size of slot numbers and settings are not matching.
- If the card does not support load-balancing.
- If there is no module present in the given slot.

Parameters

- opContext—Operational context
- networkElementId—InstanceNameId of the network element whose network card Ethernet channel load balance settings have to be changed
- networkCardSlotNos—slot numbers of the network cards whose load balance settings have to be modified. The list should contain the instances of Integer.
- settings—load balance settings to be applied. The list should contain the instances of NetworkCardEtherChannelLoadBalanceSetting

Return Value

void

modifyNetworkInterfaceLinks

Modifies an existing port and channel association attributes in the given network interface links.

This methods updates only attributes(For ex: mode) in the association between port and channel not member port associations.

Member port mode will be updated based on Ethernet channel end point protocol as follows:

- ProtocolDefault Mode
- LACPActive
- NONEOn

ValidationException is thrown if any of the following situation occurs:

- If the networkInterfaceLinks is null or empty.
- If the networkInterfaceLinks contains one or more null element, or the collection contains objects that are not of type NetworkInterfaceLink
removePortsFromEtherChannelEndPoint

Removes the given port members from the given Ethernet channel end point.

Protocol will be set to NONE, if the given Ethernet channel end point doesn't have member ports after removing given port members.

ValidationException is thrown if any of the following situation occurs:
- If channelEpId is null.
- If channelEpId is not a valid Ethernet channel end point InstanceNameId.
- If the portIds collection is null or the collection is empty or the collection contains objects other than InstanceNameId of physical ports.

PropertiesException is thrown if any of the following situation occurs:
- If given ports operates in different mode (ex : switched) than the given Ethernet channel end point (ex : routed).

IntegrityException is thrown if any of the following situation occurs:
- If the channelEpId is a SwitchedEtherChannelNetworkInterface or RoutedEtherChannelNetworkInterface InstanceNameId that does not exist in the database.
- If the portIds collection contains a NetworkInterface InstanceNameId that does not exist in the database.

Parameters
- opContext—Operational context
- portIds—list of InstanceNameId of ports (SwitchedNetworkInterface / RoutedNetworkInterface) to be removed
- channelEpId—InstanceNameId of the SwitchedEtherChannelNetworkInterface/RoutedEtherChannelNetworkInterface from which the given port members should be removed

Return Value
- void
EventApp Service

This chapter describes the DCNM web services’ API methods for the EventApp service.

Information About EventApp Service

The EventApp service provides you with various methods to obtain persistent events in the database.

createUserAction

Persist the userActions done for a list of events

Parameters:
- opContext—Operational context
- annotatedEvents—List of userActions performed on a list of events

Return Value:
void

deleteEventsBeforeATimeStamp

Deletes the events having timestamp earlier than the specified Timestamp object.

Parameters:
- opContext—Operational context
- timestamp—specifies the Timestamp

Return Value:
void

deleteEventsBetweenTimestamps

Deletes the events having timestamp between two timeInstances (startTime and endTime)
**getEventCountAfterATimestamp**

Returns the number of events having timestamp later than the specified Timestamp object. If no Events present in the Database later than specified timestamp will return 0.

**Parameters**
- opContext—Operational context
- timestamp—specifies the Timestamp

**Return Value**
The number of events having timestamp later than the specified timestamp

**getEventsAfterATimestamp**

Returns a list of Events having timestamp later than the specified Timestamp object. If no Events present in the Database later than specified timestamp will return an empty List.

**Parameters**
- opContext—Operational context
- timestamp—specifies the Timestamp
- eventCount—specifies the no.of events to be retrieved from the Database

**Return Value**
List of the Events having timestamp later than the specified timestamp

**getEventsBeforeATimestamp**

Returns a list of Events having timestamp earlier than the specified Timestamp object. If no Events present in the Database earlier to the specified timestamp, will return an empty List.

**Parameters**
- opContext—Operational context
- timestamp—specifies the Timestamp
- eventCount—specifies the no.of events to be retrieved from the Database
getEventsBetweenTimestamps

Returns a list of Events having timestamp between two timeInstances (startTime and endTime)

Parameters
opContext—Operational context
startTime—Start Timestamp
endTime—End Timestamp

Return Value
List of the Events having timestamp between two timeInstances (startTime and endTime)
If the specified startTime is null, then will return events having timestamp earlier than the endTime.
If the specified endTime is null, then will return events having timestamp later than the startTime
If the specified startTime and endTime are null, then will return all the persisted events.

getEventsBetweenTimestampsByEventClass

Returns the List of Events having timeStamp between two timeInstances (startTime and endTime) with the specified filterType.

Parameters
opContext—Operational context
startTime—Start Timestamp
endTime—End Timestamp

filterType—specifies the type of Events to be retrieved from the Database.
Can specify the following filterTypes (Constants)
- DcmNormalizedEvent
- DcmThresholdEvent
- DcmInformationalEvent
- UnsupportedEvent
- All type of Events

Return Value
List of the Events having timestamp between two timeInstances (startTime and endTime) and corresponding filterType
If the specified startTime is null, then will return events having timestamp earlier than the endTime.
If the specified endTime is null, then will return events having timestamp later than the startTime
If the specified startTime and endTime are null, then will return all the events persisted in the Database.
getEventsForEvent

Returns the List of Events having specified EventType

Parameters
opContext—Operational context
eventType—specifies the EventType

Return Value
List returns a List of Events corresponding specified eventType and filterType

getEventsForEventByEventClass

Returns the List of Events having specified EventType and filterType.

Parameters
opContext—Operational context
eventType—specifies the EventType
filterType—specifies the type of Events to be retrieved from the Database.
  Can specify the following filterTypes (Constants)
  – DcmNormalizedEvent
  – DcmThresholdEvent
  – DcmInformationalEvent
  – UnsupportedEvent
  – All type of Events

Return Value
List returns a List of Events corresponding specified eventType and filterType

getEventsForFeature

Returns the List of Events having specified FeatureType.

Parameters
opContext—Operational context
featureType—specifies the FeatureType

Return Value
List of persisted Events having specified featureType
getEventsForFeatureByEventClass

Returns the List of Events having specified FeatureType and filterType.

**Parameters**

- opContext—Operational context
- featureType—specifies the FeatureType
- filterType—specifies the type of Events to be retrieved from the Database.
  
  Can specify the following filterTypes (Constants)
  
  - DcmNormalizedEvent
  - DcmThresholdEvent
  - DcmInformationalEvent
  - UnsupportedEvent
  - All type of Events

**Return Value**

List of persisted Events having specified featureType and filterType

getEventsForPerceivedSeverity

Returns the List of Events having specified PerceivedSeverity.

**Parameters**

- opContext—Operational context
- perceivedSeverity—specifies the PerceivedSeverity

**Return Value**

List of persisted Events having specified perceivedSeverity

getEventsForPerceivedSeverityByEventClass

Returns the List of Events having specified PerceivedSeverity and filterType.

**Parameters**

- opContext—Operational context
- perceivedSeverity—specifies the PerceivedSeverity
- filterType—specifies the type of Events to be retrieved from the Database.
  
  Can specify the following filterTypes (Constants)
  
  - DcmNormalizedEvent
  - DcmThresholdEvent
  - DcmInformationalEvent
### getLastEvent

Returns the Last persisted event in the Database

**Parameters**
- **opContext**—Operational context

**Return Value**
Last persisted event in the Database

### getUserActions

Returns a List of AnnotatedEvents having specified eventId.

**Parameters**
- **opContext**—Operational context
- **eventId**—persistedEventId

**Return Value**
Returns a List of AnnotatedEvents having eventId

### modifyEventStatus

Update the EventStatus in the Database for a List of events

**Parameters**
- **opContext**—Operational context
- **eventId**—contains list of eventIds
- **eventStatusList**—contains a list of modified EventStatus

**Return Value**
void
CHAPTER 14

GlbpApp Service

This chapter describes the DCNM web services’ API methods for the GlbpApp service.

Information About GlbpApp Service

The Gateway Load Balancing Protocol (GLBP) is a first-hop routing protocol that provides gateway redundancy and load balancing. GLBP provides redundancy with an active gateway and a standby gateway. Members of a GLBP group elect one gateway to be the active virtual gateway (AVG) for that group. Other members of a GLBP group may act as redundant GLBP devices that will become active if any existing gateway fails. The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Modify APIs—Modify existing GLBP group settings.
- Remove APIs—Disassociate GLBP group setting from one or more interfaces in the device.

addGlbpGroupSetting

Adds a GLBP group setting to one or more IpNetworkInterface.

ValidationException is thrown if any of the following situation occurs:

- If ifInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If ifInstanceNameIdCol is not a valid InstanceNameId of IpNetworkInterface object.
- If setting is null or
  1. virtualIpAddress or secondaryIpAddress of a GlbpGroupSetting conflicts with the following.
     - IpAddress of IpNetworkInterface.
     - virtualIpAddress of other GlbpGroupSetting.
     - secondaryIpAddress of other GlbpGroupSetting.
     - NetworkInterface is a Voice VLAN Enabled Interface.
  2. groupName of a GlbpGroupSetting conflicts with groupName of other GlbpGroupSettings in a device

IntegrityException is thrown if any of the following situation occurs:

- If ifInstanceNameIdCol contains more than one instanceNameIds of Interfaces present in a device and
**bindTrackingObjectsToGlbpGroupSetting**

Binds an existing GlbpGroup to track a new object.

ValidationException is thrown if any of the following situation occurs:
- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.

**Parameters**
- opContext—Operational context
- objectTrackingSetting—ObjectTrackingSetting object.
- glbpInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting

**Return Value**
- void

**createGlbpGroupSettings**

Create a list of GlbpGroupSetting in different network elements.

Following associations are to be provided for the given GlbpGroupSetting:
- NetworkInterfaceReference corresponding to the specified group IpNetworkInterface reference

ValidationException is thrown if any of the following situation occurs:
- If glbpGroupSettingCol is null or empty or it is not of type GlbpGroupSetting.
- virtualIpAddress or secondaryIpAddress of a GlbpGroupSetting conflicts with the following:
  -IpAddress of IpNetworkInterface.
  -virtualIpAddress of other GlbpGroupSetting.

**Parameters**
- virtualIpAddress is not null
- groupName is not null

PropertiesException is thrown if any of the following situation occurs:
- In the glbpGroupSettingCol collection, if any one of the GlbpGroupSetting attribute is not valid.

Example:
- glbpGroupName of a GlbpGroupSetting contains a space.
disableService

Disables GlbpService in the device. Given the instance name ID of one or more network elements, GLBP will be disabled on those elements. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If neInstanceNameIdCol does not a valid AbstractNetworkElement InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameIdCol—List of InstanceNameId of GLBP disabled AbstractNetworkElement

Return Value
- void

enableService

Enables GlbpService in the device. Given the instance name ID of one or more network elements, GLBP will be enabled on those elements. This API is to enable "GLBP service" option globally in the device. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If neInstanceNameIdCol does not a valid AbstractNetworkElement InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameIdCol—List of InstanceNameId of AbstractNetworkElement

Return Value
- void
getActiveTimers

Returns a list of ActiveTimers objects corresponding to one or more GLBP groups. Given a list of InstanceNameId corresponding to GlbpGroupSetting, API returns a list of ActiveTimers corresponding to the GLBP groups.

ValidationException is thrown if any of the following situation occurs:

- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.

Parameters

- opContext—Operational context
- glbpSettingInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting

Return Value

List of ActiveTimers objects

ggetAllGlbpGroupSettings

Returns a list of GlbpGroupSetting corresponding to GLBP groups configured on all the managed network elements.

Parameters

- opContext—Operational context

Return Value

List of GlbpGroupSetting objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- IP network interface corresponding to GlbpGroupSetting
- GLBP Gateway Status
- GLBP Forwarder Status

ggetAllGlbpGroupsInNetwork

Returns all GLBP groups configured in a network. Returns a list of GlbpGroup. GlbpGroup contains the GLBP group ID and a collection of GlbpGroupSetting corresponding the GLBP group members.

Parameters

- opContext—Operational context
Return Value
List of GlbpGroup. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- IP network interface corresponding to GlbpGroupSetting
- GLBP Gateway Status
- GLBP Forwarder Status

getForwarderStatus

Returns a list of GlbpForwarderStatus for a specified GLBP group on a network element. Given the InstanceNameId of GlbpGroupSetting, returns a list of GlbpForwarderStatus.
ValidationException is thrown if any of the following situation occurs:
- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.

Parameters
opContext—Operational context
glbpSettingInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting

Return Value
List of GlbpForwarderStatus. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- GLBP group setting corresponding to GlbpForwarderStatus

getGlbpAuthenticationSetting

Returns a list of GlbpAuthenticationSetting corresponding to one or more GLBP groups. Given a list of InstanceNameId corresponding to GlbpGroupSetting, API returns a list of GlbpAuthenticationSetting corresponding to the GLBP groups.
ValidationException is thrown if any of the following situation occurs:
- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.

Parameters
opContext—Operational context
glbpSettingInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting

Return Value
List of GlbpAuthenticationSetting objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- GLBP group setting corresponding to GlbpAuthenticationSetting
- Key Chain
getGlbpGatewayStatus

Returns a list of GlbpGatewayStatus corresponding to one or more GLBP groups. Given a list of InstanceNameId corresponding to GlbpGroupSetting, API returns a list of GlbpGatewayStatus corresponding to the GLBP groups.

ValidationException is thrown if any of the following situation occurs:

- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.

**Parameters**

- opContext—Operational context
- glbpSettingInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting

**Return Value**

List of GlbpGatewayStatus.

getGlbpGlobalSetting

Returns a list of GlbpGlobalSetting objects.

**Parameters**

- opContext—Operational context
- neInstanceNameIdCol—InstanceNameId of the network element

**Return Value**

List of GlbpGlobalSetting objects.

getGlbpGroupIdsInNetworkInterface

Returns a list of groupIds of GlbpGroupSetting configured in an IpNetworkInterface

**Parameters**

- opContext—Operational context
- ifInstanceNameIdCol—InstanceNameId of IpNetworkInterface

**Return Value**

List of Group Ids of GlbpGroupSetting

getGlbpGroupSetting

Returns a list of GlbpGroupSetting objects corresponding to a list of InstanceNameId.

ValidationException is thrown if any of the following situation occurs:
getGlbpGroupSettingsInNetworkElement

Returns a list of GlbpGroupSetting corresponding to GLBP groups configured on a network element. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null or it is not a valid InstanceNameId of a network element.

**Parameters**
- opContext—Operational context
- neInstanceNameId—InstanceNameId of the network element

**Return Value**
List of GlbpGroupSetting objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared:

- IP network interface corresponding to GlbpGroupSetting
- GLBP Gateway Status
- GLBP Forwarder Status

getGlbpGroupSettingsInNetworkInterface

Returns a list of groupIds of GlbpGroupSetting configured in an IpNetworkInterface

**Parameters**
- opContext—Operational context
- ifInstanceNameIdCol—InstanceNameId of IpNetworkInterface

**Return Value**
List of GroupSettings configured in the interfaces
getGlbpTimers

Returns a list of GlbpTimers objects corresponding to one or more GLBP groups. Given a list of InstanceNameId corresponding to GlbpGroupSetting, API returns a list of GlbpTimers corresponding to the GLBP groups.

ValidationException is thrown if any of the following situation occurs:

- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.

Parameters

- opContext—Operational context
- glbpSettingInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting

Return Value

List of GlbpTimers objects

getGlbpVirtualForwarderSetting

Returns a list GlbpForwarderSetting for a GLBP group on a network element. Given the InstanceNameId of GlbpGroupSetting, returns a list of GlbpForwarderSetting corresponding to the group.

ValidationException is thrown if any of the following situation occurs:

- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.

Note:

- Forwarder Id, Forwarder Priority and ForwarderMacAddress cannot be configured in Cat-6k

Parameters

- opContext—Operational context
- glbpSettingInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting

Return Value

List of GlbpForwarderSetting. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- GLBP group setting corresponding to GlbpForwarderSetting

getGlbpWeightingSetting

Returns a list of GlbpWeightingSetting corresponding to one or more GLBP groups. Given a list of InstanceNameId corresponding to GlbpGroupSetting, API returns a list of GlbpWeightingSetting corresponding to the GLBP groups.

ValidationException is thrown if any of the following situation occurs:

- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
getTrackedObjects

Returns a list of objects tracked by a GLBP group. Given the InstanceNameId corresponding to GlbpGroupSetting, returns a list of GlbpGroupTracksNetworkInterface.

ValidationException is thrown if any of the following situation occurs:
- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.

Parameters
- opContext—Operational context
- glbpSettingInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting

Return Value
List of GlbpGroupTracksNetworkInterface objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- ObjectTrackingSetting can be obtained using GlbpGroupTracksNetworkInterface. The network interface being tracked can be obtained from the associated ObjectTrackingSetting.

modifyAuthenticationSetting

Modifies GlbpAuthenticationSetting for one or more GlbpGroupSetting. KeyChain associations can also be built with GlbpAuthenticationSetting.

ValidationException is thrown if any of the following situation occurs:
- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.
- If setting is null.

Parameters
- opContext—Operational context
- glbpSettingInstanceNameIdCol—list of InstanceNameId of GlbpGroupSetting
- glbpTimers—GLBP authentication setting to be applied to GLBP groups.

Return Value
void
modifyGlbpGroupSetting

Modifies GlbpGroupSetting on one or more network elements.

ValidationException is thrown if any of the following situation occurs:

- If glbpGroupSettingCol is null or empty or it is not of type GlbpGroupSetting.
- Only existing GlbpGroupSetting (exist in database) can be modified.
- virtualIpAddress or secondaryIpAddress of a GlbpGroupSetting conflicts with the following.
- IpAddress of IpNetworkInterface.
- virtualIpAddress of other GlbpGroupSetting.
- secondaryIpAddress of other GlbpGroupSetting.
- NetworkInterface is a Voice VLAN Enabled Interface.
- groupName of a GlbpGroupSetting conflicts with groupName of other GlbpGroupSettings in a device

PropertiesException is thrown if any of the following situation occurs:

- In the glbpGroupSettingCol collection, if any one of the GlbpGroupSetting attribute is not valid.

Example:

- glbpGroupName of a GlbpGroupSetting contains a space.

Parameters

opContext—Operational context

glbpGroupSettingCol—List of modified GlbpGroupSetting objects.

Return Value

void

modifyTimers

Modifies GlbpTimers for one or more GlbpGroupSetting.

ValidationException is thrown if any of the following situation occurs:

- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.
- If glbpTimers is null.

Parameters

opContext—Operational context

glbpSettingInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting

glbpTimers—GLBP timer setting to be applied to GLBP groups.

Return Value

void
modifyWeightingSetting

Modifies GlbpWeightingSetting for one or more GlbpGroupSetting.

ValidationException is thrown if any of the following situation occurs:

- If glbpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If glbpSettingInstanceNameIdCol is not a valid InstanceNameId of GlbpGroupSetting object.
- If setting is null.

Parameters

opContext—Operational context
glbpSettingInstanceNameIdCol—List of InstanceNameId of GlbpGroupSetting
setting—GLBP weighting setting to be applied to GLBP groups.

Return Value

void

removeGlbpGroupSetting

Removes all GlbpGroupSetting for one or more IpNetworkInterface.

ValidationException is thrown if any of the following situation occurs:

- If ifInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If ifInstanceNameIdCol is not a valid InstanceNameId of IpNetworkInterface object.

Parameters

opContext—Operational context
ifInstanceNameIdCol—List of InstanceNameId of IpNetworkInterface

Return Value

void

removeGlbpGroupSettingByGroupId

Removes GlbpGroupSetting for a GLBP group ID on one or more NetworkInterface.

ValidationException is thrown if any of the following situation occurs:

- If glbpGroupId is null
- If ifInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If ifInstanceNameIdCol is not a valid InstanceNameId of IpNetworkInterface object.

Parameters

opContext—Operational context
glbpGroupId—GLBP group ID
unbindTrackingObjectsFromGlbpGroupSetting

Unbinds a GLBP group from the specified tracked objects so that the status of the group is no longer influenced by the status of the tracked objects.

ValidationException is thrown if any of the following situation occurs:

- glbpInstanceNameId is not a valid InstanceNameId of GlbpGroupSetting.
- If objectTrackingInstanceNameIdCol is not a valid InstanceNameId of ObjectTrackingSetting object.

Parameters

- opContext—Operational context
- glbpInstanceNameId—InstanceNameId of GlbpGroupSetting
- objectTrackingInstanceNameIdCol—List of InstanceNameId of ObjectTrackingSetting

Return Value

void
HsrpApp Service

This chapter describes the DCNM web services’ API methods for the HsrpApp service.

Information About HsrpApp Service

The Hot Standby Router Protocol (HSRP) allows a transparent failover at the first-hop IP router. The protocol allows hosts to appear to use a single router and to maintain connectivity even if the actual first-hop router fails. HSRP enables two or more devices to work together in a group to provide an illusive virtual router service by sharing a virtual IP address and virtual MAC address. Through election (based on priority), one router (having the highest priority) in the HSRP group is elected as active and another router is elected as standby. Remaining routers will be in listening mode.

APIs are defined with the following categories:

1. Query/Get APIs—Used to query data from the persisted database.
2. Modify APIs—Used to modify existing HSRP group settings.
3. Remove APIs—Used to disassociate HSRP group settings from one or more interfaces in the device.

addHsrpGroupSetting

Adds a HSRP group setting to one or more IpNetworkInterface.

ValidationException is thrown if any of the following situation occurs:

- If ifInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If ifInstanceNameIdCol is not a valid InstanceNameId of IpNetworkInterface object.
- If setting is null
- If virtualIpAddress or secondaryIpAddress of a HsrpGroupSetting conflicts with the following.
  - IpAddress of IpNetworkInterface.
  - virtualIpAddress of other HsrpGroupSetting.
  - secondaryIpAddress of other HsrpGroupSetting.
- If groupName of a HsrpGroupSetting conflicts with groupName of other HsrpGroupSettings in a device

IntegrityException is thrown if any of the following situation occurs:
bindTrackingObjectToHsrpGroupSetting

Binds an existing HsrpGroup to track a new object.
ValidationException is thrown if any of the following situation occurs:
• If hsrpInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
• If objectTrackingInstanceNameIdCol is not a valid InstanceNameId of ObjectTrackingSetting object.

Parameters
opContext—Operational context
hsrpInstanceNameIdCol—List of InstanceNameId of HsrpGroupSetting
objectTrackingInstanceNameIdCol—List of InstanceNameId of ObjectTrackingSetting object.

decrementValue—Decrement Value

Return Value
void

disableHsrpService

Disables HsrpService in the device. Given the instance name ID of one or more network elements, hsrp will be disabled on those elements. ValidationException is thrown if any of the following situation occurs:
• If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
enableHsrpService

Enables HsrpService in the device. Given the instance name ID of one or more network elements, hsrp will be enabled on those elements. This API is to enable "hsrp service" option globally in the device.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If neInstanceNameIdCol does not a valid AbstractNetworkElement InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—List of InstanceNameId of AbstractNetworkElement

Return Value

void

getHsrpAuthenticationSetting

Returns a list of HsrpAuthenticationSetting corresponding to one or more HSRP groups. Given a list of InstanceNameId corresponding to HsrpGroupSetting, API returns a list of HsrpAuthenticationSetting corresponding to the HSRP groups.

ValidationException is thrown if any of the following situation occurs:
- If hsrpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If hsrpSettingInstanceNameIdCol does not a valid InstanceNameId of HsrpGroupSetting object.

Parameters

- opContext—Operational context
- hsrpSettingInstanceNameIdCol—List of InstanceNameId of HsrpGroupSetting

Return Value

List of HsrpAuthenticationSetting objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- HSRP group setting corresponding to HsrpAuthenticationSetting
- Key Chain
getHsrpGlobalSetting

Returns a list of (HsrpGlobalSetting) objects. Each (HsrpGlobalSetting) object will have a Boolean attribute hsrpServiceEnable populated which indicates whether HSRP service is enabled/disabled on this device.

**Parameters**
- opContext—Operational context
- neInstanceNameIdCol—

**Return Value**
List of HsrpGlobalSetting objects.

getHsrpGroupSetting

Returns a list of HsrpGroupSetting objects corresponding to a list of InstanceNameId. ValidationException is thrown if any of the following situation occurs:

- If hsrpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If hsrpSettingInstanceNameIdCol is not a valid InstanceNameId of HsrpGroupSetting object.

**Parameters**
- opContext—Operational context
- hsrpSettingInstanceNameIdCol—List of InstanceNameId corresponding to HsrpGroupSetting

**Return Value**
List of HsrpGroupSetting objects with all the associations populated.

getHsrpGroupSettingsInNetworkElement

Returns a list of HsrpGroupSetting corresponding to HSRP groups configured on a network element. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null or it is not a valid InstanceNameId of a network element.

**Parameters**
- opContext—Operational context
- neInstanceNameId—InstanceNameId of the network element

**Return Value**
List of HsrpGroupSetting objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- IP network interface corresponding to HsrpGroupSetting
- HSRP Router Status
getHsrpRouterStatus

Returns a list of HsrpRouterStatus corresponding to one or more HSRP groups. Given a list of InstanceNameId corresponding to HsrpGroupSetting, API returns a list of HsrpRouterStatus corresponding to the HSRP groups.

ValidationException is thrown if any of the following situation occurs:
- If hsrpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If hsrpSettingInstanceNameIdCol is not a valid InstanceNameId of HsrpGroupSetting object.

Parameters
- opContext—Operational context
- hsrpSettingInstanceNameIdCol—List of InstanceNameId of HsrpGroupSetting

Return Value
List of HsrpRouterStatus.

getHsrpTimers

Returns a list of HsrpTimersSetting objects corresponding to one or more HSRP groups. Given a list of InstanceNameId corresponding to HsrpGroupSetting, API returns a list of HsrpTimersSetting corresponding to the HSRP groups.

ValidationException is thrown if any of the following situation occurs:
- If hsrpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If hsrpSettingInstanceNameIdCol is not a valid InstanceNameId of HsrpGroupSetting object.

Parameters
- opContext—Operational context
- hsrpSettingInstanceNameIdCol—List of InstanceNameId of HsrpGroupSetting

Return Value
List of HsrpTimersSetting objects.

getIpInterfaceWithHsrpSettings

Returns a list of IpNetworkInterface. Given a list of InstanceNameId corresponding to IpNetworkInterface, API returns a list of IpNetworkInterface with HsrpNetworkInterfaceSetting populated.

ValidationException is thrown if any of the following situation occurs:
- If ipInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If ipInstanceNameIdCol is not a valid InstanceNameId of IpNetworkInterface object.

Parameters
- opContext—Operational context
getTrackingObject

Returns a list of HsrpGroupTracksObject corresponding to one or more HSRP groups. Given a list of InstanceNameId corresponding to HsrpGroupSetting, API returns a list of HsrpGroupTracksObject corresponding to the HSRP groups.

ValidationException is thrown if any of the following situation occurs:

- If hsrpSettingInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If hsrpSettingInstanceNameIdCol is not a valid InstanceNameId of HsrpGroupSetting object.

Parameters

- opContext—Operational context
- hsrpSettingInstanceNameIdCol—List of InstanceNameId of HsrpGroupSetting

Return Value

List of HsrpGroupTracksObject.

modifyAuthenticationSetting

Modifies HsrpAuthenticationSetting for one or more HsrpGroupSetting. KeyChain associations can also be built with HsrpAuthenticationSetting.

ValidationException is thrown if any of the following situation occurs:

- If hsrpInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If hsrpInstanceNameIdCol is not a valid InstanceNameId of HsrpGroupSetting object.
- If setting is null.

Parameters

- opContext—Operational context
- hsrpInstanceNameIdCol—List of InstanceNameId of HsrpGroupSetting
- setting—HSRP authentication setting to be applied to HSRP groups.

Return Value

void

modifyHsrpGroupSetting

Modifies HsrpGroupSetting on one or more network elements.

ValidationException is thrown if any of the following situation occurs:
modifyHsrpTimers

Modifies HsrpTimersSetting for one or more HsrpGroupSetting.

ValidationException is thrown if any of the following situation occurs:

- If hsrpInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If hsrpInstanceNameIdCol is not a valid InstanceNameId of HsrpGroupSetting object.
- If hsrpTimers is null.

Parameters

opContext—Operational context
hsrpInstanceNameIdCol—List of InstanceNameId of HsrpGroupSetting
hsrpTimers—HSRP timer setting to be applied to HSRP groups.

Return Value

List of updated HsrpTimersSetting objects

modifyIpInterfaceWithHsrpSettings

Given a list of InstanceNameId corresponding to IpNetworkInterface and a
HsrpNetworkInterfaceSettingobject, API modifies the HsrpNetworkInterfaceSetting associated with a
IpNetworkInterface ValidationException is thrown if any of the following situation occurs:

- If hsrpGroupSettingCol is null or empty or it is not of type HsrpGroupSetting.
- Only existing HsrpGroupSetting (exist in database) can be modified.
- virtualIpAddress or secondaryIpAddressof a HsrpGroupSetting conflicts with the following.
  1. IpAddress of IpNetworkInterface.
  2. virtualIpAddress of other HsrpGroupSetting.
  3. secondaryIpAddress of other HsrpGroupSetting.
- groupName of a HsrpGroupSetting conflicts with groupName of other HsrpGroupSettings in a device

PropertiesException is thrown if any of the following situation occurs:

- In the hsrpGroupSettingCol collection, if any one of the HsrpGroupSetting attribute is not valid.

Example:

1. hsrpGroupName of a HsrpGroupSetting contains a space.

Parameters

opContext—Operational context
hsrpGroupSettingCol—List of modified HsrpGroupSetting objects.

Return Value

List of HsrpGroupSetting
removeHsrpGroupSetting

Removes a list of Hsrp Groups HsrpGroupSetting for a IpNetworkInterface.

Parameters
opContext—Operational context
hsrpGroupIdCol—List of InstanceNameId of HsrpGroupSetting
ifInstanceNameId—InstanceNameId of the IpNetworkInterface object

Return Value
void

unbindTrackingObjectFromHsrpGroupSetting

Unbinds a HSRP group from the specified tracked objects so that the status of the group is no longer influenced by the status of the tracked objects.

Parameters
opContext—
hsrpInstanceNameIdCol—List of InstanceNameId of HsrpGroupSetting
objectTrackingInstanceNameIdCol—List of InstanceNameId of ObjectTrackingSetting

Return Value
void
CHAPTER 16

InterfacesApp Service

This chapter describes the DCNM web services’ API methods for the InterfacesApp service.

Information About InterfacesApp Service

This chapter defines the APIs exposed by the Interfaces service feature.

createLoopbackNetworkInterfaces

Creates one or more Loopback interface objects in a network element. Given the instance name Id of network element and list of loopback interface objects, creates the objects in the server and returns its instance name Ids. No associations will be considered for the given object while creating the interface in the server.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the loopBackIns is null or the collection is empty.
- If the loopBackIns contains one or more null element, or the collection contains objects that are not of type LoopBackNetworkInterface.

PropertiesException is thrown if any of the following situation occurs:

- In the loopBackIns collection, if any of the LoopBackNetworkInterface attribute is not valid.

Example:

- Configuring attribute description with a String of length more than 240 will cause an PropertiesException, because description of an interface can only be up to a maximum of 240 characters.

IntegrityException is thrown incase of any of the LoopBackNetworkInterface already exists in the database.

Parameters

- opContext—Operational context
- neInstanceNameId—Instance name ID of the network element.
- loopbackIns—loopback objects that need to be created.
createRoutedSubNetworkInterfaces

Create one or more routed sub-interfaces object for a routed physical interface. Given the instance name Id of routed interface and list of routed sub-network interfaces, creates the objects in the server and returns its instance name Ids.

Following associations will be updated for the given interfaces in the server:

- Encapsulation VLAN

ValidationException is thrown if any of the following situation occurs:

- If niId is null.
- If niId is not a valid InstanceNameId.
- If the subInfs is null or the collection is empty.
- If the subInfs contains one or more null element, or the collection contains objects that are not of type RoutedSubNetworkInterface.
- If there is no object existing in database corresponding to the InstanceNameId.
- If IP address (Ipv4AddressPrefix) being configured to the interface without creating a encapsulation VLAN (VLAN) association
- If the configured IP address (Ipv4AddressPrefix) overlaps with primary or secondary IP address of any other Layer 3 interfaces existing in the device.
- If the configured encapsulation VLAN (VLAN) is not available or already in use by any other interfaces.
- If encapsulation type (SubInterfaceEncapsulationType) is configured without specifying encapsulation VLAN.

PropertiesException is thrown if any of the following situation occurs:

- In the subInfs collection, if any of the RoutedSubNetworkInterface attribute is not valid.

Example:

- Configuring attribute description with a String of length more than 240 will cause an PropertiesException, because description of an interface can only be up to a maximum of 240 characters.

IntegrityException is thrown incase of any of the RoutedSubNetworkInterface already exists in the database.

Parameters

- opContext—Operational context
- niId—instance name Id of RoutedNetworkInterface
- subInfs—List of RoutedSubNetworkInterface objects that need to be created

Return Value

Instance name Ids of newly created routed sub-network interface objects.
createVlanNetworkInterfaces

Creates one or more VLAN interface (SVI) objects in a network element. Given the instance name Id of network element and list of SVI objects, creates the objects in the server and returns its instance nameIds. No associations will be considered for the given object while creating the interface in the server.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the vlanIns is null or the collection is empty.
- If the vlanIns contains one or more null element, or the collection contains objects that are not of type VlanNetworkInterface.

PropertiesException is thrown if any of the following situation occurs:
- In the vlanIns collection, if any of the VlanNetworkInterface attribute is not valid.

Example:
- Configuring attribute description with a String of length more than 240 will cause an PropertiesException, because description of an interface can only be up to a maximum of 240 characters.

IntegrityException is thrown incase of any of the VlanNetworkInterface already exists in the database.

Parameters
- opContext—Operational context
- neInstanceNameId—Instance name ID of the network element.
- vlanIns—SVI objects that need to be created.

Return Value
Instance name IDs of the newly created SVI objects.

deleteLogicalNetworkInterfaces

Delete one or more existing layer 3 logical network interface objects. This method can be used for the following logical interfaces:
a) Loopback interfaces
b) SVI
c) Routed sub-network interface

ValidationException is thrown if any of the following situation occurs:
- If niInstanceNameIds collection is null or it is empty.
- If niInstanceNameIds collection contains an element that is not of type IpNetworkInterface InstanceNameId.
- If niInstanceNameIds collection contains a NetworkInterface that does not exist in the database.

Deletion of both RoutedEtherChannelNetworkInterface and IpGreTunnelNetworkInterface are not supported in this method.
disableSviService

Disables SVI service for the given device Ids.

Parameters
opContext—Operational context
niInstanceNameIds—list of instance name Ids of the NetworkInterface objects.

Return Value
void

disableSviService

enableSviService

Parameters
opContext—Operational context
neInstanceNameIds—InstanceOfId corresponding to the network elements

Return Value
void

enableSviService

Enables SVI service for the given device Ids.

Parameters
opContext—Operational context
neInstanceNameIds—InstanceOfId corresponding to the network elements

Return Value
void

Error Conditions
ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null.
- If the neInstanceNameIdCol collection is empty or the collection contains object which are not a valid network element InstanceNameId
enableUdldService

Enables UDLD service for the given device Ids.

ValidationException is thrown if any of the following situation occurs:
- If the argument passed is null.
- If the neInstanceNameIdCol collection is empty or the collection contains object which are not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameIds—InstanceNameId corresponding to the network elements

Return Value
- void

fetchCurrentTransceiverStatusForPorts

Returns a list of network interface object populated with all transceiver status associations.

ValidationException is thrown if the argument passed is null or it is not a valid network interface InstanceNameId.

Parameters
- opContext—Operational context
- interfaceIdCol—InstanceNameId corresponding to the NetworkInterface

Return Value
- List of NetworkInterface instances with security associations populated.

Following associations will be populated for each interface if it is exist for the given interface:
1. sfpStatusRef
2. sfpDiagnosticsThresholdStatusRef
3. networkInterfaceNameRef
getAclAssociations

Return a collection of interface to ACL association objects corresponding to a given interface object. ValidationException is thrown if the argument passed is null or it is not a valid network interface InstanceNameId.

Parameters
- opContext—Operational context
- interfaceId—InstanceNameId corresponding to the NetworkInterface

Return Value
Set of AclAppliesToNetworkInterface objects associated to given network interface
Following objects will be populated on other end of AclAppliesToNetworkInterface association:
- MacAccessControlList
- AccessControlList

getActiveRoutedPortsCountInNetworkElement

Returns number of active routed ports present in the device
ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element

Return Value
Number of ports

getActiveSwitchedPortsCountInNetworkElement

Returns number of active switched ports present in the device for a given switch port mode
ValidationException is thrown if any of the following situation occurs:
- If the neInstanceNameId is null or it is not a valid network element InstanceNameId.
- If the mode is null

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element
- mode—Switched network interface mode
getAdminDownRoutedPortsCountInNetworkElement

Returns number of administration down routed ports present in the device
ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element

Return Value
Number of ports

getAdminDownSwitchedPortsCountInNetworkElement

Returns number of administration down switched ports present in the device for a given switch port mode
ValidationException is thrown if any of the following situation occurs:
- If the neInstanceNameId is null or it is not a valid network element InstanceNameId.
- If the mode is null

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element
- mode—Switched network interface mode

Return Value
Number of ports

getAllInterfacesInNetworkElement

Returns all types of interfaces present in a network element. Given the instance name ID of a network element, returns a collection of interface objects (both physical and logical interfaces).
ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network elements
**getErrorDetectionSettings**

Return a collection of port error detection setting for a list of network elements.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null.
- If the `neInstanceNameIds` collection is empty or the collection contains object which are not a valid network element `InstanceNameId`

**Parameters**

- `opContext`—Operational context
- `neInstanceNameIds`—`InstanceNameId` corresponding to the network elements

**Return Value**

List of `PortErrorDisableDetectionSetting` for the given list of devices. No other associations will be available.
get_ErrorRecoverySettings

Return a collection of port error recovery setting for a list of network elements.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null.
- If the neInstanceNameIds collection is empty or the collection contains object which are not a valid network element InstanceNameId

Parameters

- opContext—Operational context
- neInstanceNameIds—InstanceNameId corresponding to the network elements

Return Value

List of PortErrorDisableRecoverySetting for the given list of devices. No other associations will be available.

getIpAclAssociationsForNetworkInterface

Returns collection of interface to IP ACL association objects corresponding to a given interface object.

ValidationException is thrown if the argument passed is null or it is not a valid network interface InstanceNameId.

Parameters

- opContext—Operational context
- interfaceId—InstanceNameId corresponding to the RoutedNetworkInterface

Return Value

List of AclAppliesToNetworkInterface objects associated to given network interface

Following objects will be populated on other end of AclAppliesToNetworkInterface association:

- AccessControlList

getLogicalInterfacesInNetworkElement

Returns all the logical interfaces present in a network element. Given the instance name ID of a network element, returns a collection of logical interface objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of network element
getLoopbackInterfacesInNetworkElement

Returns all the Loopback interfaces present in a network element. Given the instance name ID of a network element, returns a collection of LoopBack interface objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId of network element

Return Value
List of all the LoopBackNetworkInterface present in the network element
Following associations will be available for each interface:
• NetworkInterfaceName
• NetworkInterfaceStatus

getMacAclAssociationsForNetworkInterface

Returns collection of interface to MAC ACL association objects corresponding to a given interface object.

ValidationException is thrown if the argument passed is null or it is not a valid network interface InstanceNameId.

Parameters
opContext—Operational context
interfaceId—InstanceNameId corresponding to the SwitchedNetworkInterface

Return Value
List of AclAppliesToNetworkInterface objects associated to given network interface
Following objects will be populated on other end of AclAppliesToNetworkInterface association:
• MacAccessControlList
getMissingPortGroupMembers

Returns a list of port group members that are missed in the given list of Interface InstanceNameIds. Returns empty list when the given list has all the members of port groups or the given interfaces are not a member of a port group.

ValidationException is thrown if any of the following situations occurs:

- If interfaceIds is null or it is empty.
- If interfaceIds contains invalid NetworkInterface InstanceNameId or null value.
- If there is no equivalent NetworkInterface object with the given InstanceNameId in the interfaceIds.

**Parameters**

opContext—Operational context
interfaceIds—A List of InstanceNameId objects of SwitchedNetworkInterface and RoutedNetworkInterface objects.

**Return Value**

Returns a List of or RoutedNetworkInterface objects.

In the returned Interface objects only the following associations will be available and all other associations will be cleared.

1. NetworkInterfaceName
2. NetworkInterfaceStatus
3. PortCapability

getMgmtInterfaces

Returns the MgmtNetworkInterface for a given list of device. Returned list may have null entries if any of the NetworkElement corresponding to neInstanceNameIdCol has no MgmtNetworkInterface. Ex: Catalyst 6500 series switches devices has no MgmtNetworkInterface. If null or empty list is passed for neInstanceNameIdCol, then MgmtNetworkInterface of all Nexus 7000 series switch devices will be returned.

- If neInstanceNameIdCol contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIdCol.

**Parameters**

opContext—Operational context
neInstanceNameIdCol—InstanceNameId corresponding to the network elements

**Return Value**

List of management interfaces present in the device.
getNetworkInterfaceSettings

Returns a collection of network interface global settings for a given list of device Ids.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null.
- If the neInstanceNameIds collection is empty or the collection contains object which are not a valid network element InstanceNameId

Parameters

opContext—Operational context
neInstanceNameIds—InstanceNameId corresponding to the network elements

Return Value

List of NetworkInterfaceSetting for the given list of devices. No association will be available.

getNetworkInterfaces

Returns a collection of network interface objects for the given list of interface Ids.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null.
- If the interfaceIds collection is empty or the collection contains object which are not a valid network interface InstanceNameId

Parameters

opContext—Operational context
interfaceIds—instance ids of list interface for which the information is required

Return Value

List of instance ids of NetworkInterface NetworkInterface instance

Following associations will be available in case of physical port:

- NetworkInterfaceName
- NetworkInterfaceStatus
- PortSetting
- PortCapability
- PortStatus
- Routed sub-network interfaces in case of routed port

Following associations will be available for routed sub-network interface:

- NetworkInterfaceName
- NetworkInterfaceStatus
- Encapsulation VLAN

Following associations will be available in case of logical interface (PortChannel, SVI, Loopback):
getNetworkInterfaceByName

Returns the List of NetworkInterface objects given the names or range of the NetworkInterface.
ValidationException is thrown if the neInstanceNameId passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element
- interfaceRange—Interface names or range to be returned.

Return Value
List of NetworkInterface objects corresponding to the interfaceRange provided. If there is no NetworkInterface object with the name given, ignores it.

getOperDownRoutedPortsCountInNetworkElement

Returns number of operationally down routed ports present in the device
ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element

Return Value
Number of ports

getOperDownSwitchedPortsCountInNetworkElement

Returns number of operationally down switched ports present in the device for a given switch port mode
ValidationException is thrown if any of the following situation occurs:
- If the neInstanceNameId is null or it is not a valid network element InstanceNameId.
- If the mode is null

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element
- mode—switched network interface mode
getPortStatusSummary

Returns status summary for both Switched and Routed ports, here status summary includes count of ports which are operationally up/down, administration up/down and the total count of switched/routed ports.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null.
- If the neInstanceNameIds collection is empty or the collection contains object which are not a valid network element InstanceNameId

Parameters

opContext—Operational context
neInstanceNameIds—InstanceNameId corresponding to the network elements

Return Value

A List of

getPortUsedForDiscovery

Returns a instance of network interface with IP address equal to the management IP address (either IPv4 or IPv6 primary address) of the network element corresponding to neInstanceNameId.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameId is null

Parameters

opContext—Operational context
neInstanceNameId—instance id of NetworkElement.

Return Value

A IpNetworkInterface instance.

Following associations will be available

- NetworkInterfaceName
- Ipv6 Address Setting collection

getPortsInModule

Returns all the physical ports present in the module for a given slot. Given the instance name ID of a network element and the slot number of a module, returns a collection of physical interface objects corresponding to the module.
getPortsInNetworkElement

Returns a collection of switched and routed physical interface objects present in a network element. Given the instance name ID of a network element, returns a collection of interface objects. ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceId—InstanceId corresponding to the network elements

Return Value
List of SwitchedNetworkInterface instances and RoutedNetworkInterface instances present in the device. In the returned list of objects only the following associations will be available and all other associations will be cleared.
- NetworkInterfaceName
- NetworkInterfaceStatus
- PortSetting
- PortCapability
- PortStatus

getPortsInNetworkElement

ValidationException is thrown if the argument passed is null or it is not a valid routed port InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceId—InstanceId corresponding to the network elements
- slotNo—slot number of the module whose ports are required

Return Value
List of SwitchedNetworkInterface instances and RoutedNetworkInterface instances present in a module. Following associations will be available:
- NetworkInterfaceName
- NetworkInterfaceStatus
- PortSetting
- PortCapability
- PortStatus
- Routed sub-network interfaces in case of routed port

Following associations will be available for routed sub-network interface:
- NetworkInterfaceName
- NetworkInterfaceStatus
- Encapsulation
getPortsWithPortChannelAssociations

Returns the network interface objects with port-channel associations populated.

ValidationException is thrown if any of the following situation occurs:

- If interfaceIds is null or it is empty.
- If interfaceIds contains invalid NetworkInterface InstanceNameId or null value.
- If there is no equivalent NetworkInterface object with the given InstanceNameId in the interfaceIds.

Parameters

- opContext—Operational context
- interfaceIds—A List of InstanceNameId objects of SwitchedNetworkInterface and RoutedNetworkInterface objects for which the port-channel associations are required.

Return Value

Returns a List of SwitchedNetworkInterface and RoutedNetworkInterface objects populated with port-channel associations

Following associations will be populated for each interface:

- Switched EtherChannel setting if the port is in switched mode
- Routed EtherChannel setting if the port is in routed mode

getPortsWithSpanAssociations

Returns network interface object with SPAN associations populated for a given network interface.

ValidationException is thrown if the argument passed is null or it is not a valid network interface InstanceNameId.

Parameters

- opContext—Operational context
- interfaceIdCol—InstanceNameId corresponding to the NetworkInterface

Return Value

Instance of NetworkInterface with span associations populated

Following associations will be populated for each interface:

- Local SPAN to source interfaces
getPortsWithVlanAssociations

Return a switched network interface object populated with VLAN associations. 
ValidationException is thrown if the argument passed is null or it is not a valid switched network interface instance ID.

Parameters
- opContext—Operational context
- interfaceIdCol—InstanceNameId corresponding to the SwitchedNetworkInterface

Return Value
Instance of SwitchedNetworkInterface

Following associations will be populated for the SwitchedNetworkInterface:
- SwitchedNetworkInterfaceBelongsToVlan which includes access VLAN, Native VLAN, Voice VLAN
- PromiscuousPvlanMapping Primary and corresponding secondary VLANs

Following associations will be populated for each VLAN which are available as part of above associations:
- VLAN collection incase of Primary VLAN
- VLAN reference in case of Secondary VLAN

getRoutedPortsInModule

Returns all the routed physical ports present in the module for a given slot. Given the instance name ID of a network element and the slot number of a module, returns a collection of routed interface objects corresponding to the specified module.

ValidationException is thrown if the argument passed is null or it is not a network element instance ID.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network elements
- slotNo—slot number of the module whose ports are required

Return Value
List of RoutedNetworkInterface instances present in a module

Following associations will be available:
- NetworkInterfaceName
getRoutedPortsInNetworkElement

Returns all the routed physical ports present in the device. Given the instance name ID of a network element, returns a collection of routed interface objects present in the device.

ValidationException is thrown if the argument passed is null or it is not a network element InstanceNameId.

Parameters

opContext—Operational context
neInstanceNameId—InstanceNameId corresponding to the network elements

Return Value

Set of RoutedNetworkInterface instances present in a module

Following associations will be available:

- NetworkInterfaceName
- NetworkInterfaceStatus
- PortSetting
- PortCapability
- PortStatus
- Routed sub-network interfaces in case of routed port

Following associations will be available for routed sub-network interface:

- NetworkInterfaceName
- NetworkInterfaceStatus
- Encapsulation VLAN

getRoutedPortsWithSecurityAssociations

Return a routed network interface object populated with all security associations.

ValidationException is thrown if the argument passed is null or it is not a valid routed network interface InstanceNameId.
Send document comments to nexus7k-docfeedback@cisco.com

Parameters
opContext—Operational context
interfaceIdCol—InstanceNameId corresponding to the RoutedNetworkInterface

Return Value
Instance of RoutedNetworkInterface with security associations populated.
Following associations will be populated for each interface:
- CtsNetworkInterfaceSetting
- Dot1xNetworkInterfaceSetting

getRoutedSubNetworkInterfacesInNetworkElement

Returns all the routed sub-interfaces objects present in a network element. Given the instance name ID of a network element, returns a collection of routed sub-interface objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId of network element

Return Value
void

getSubInterfacesForRoutedNetworkInterface

Returns all the routed sub interface objects that are associated to a routed port. Given the instance name ID of a physical routed port, returns a collection of routed sub-interface objects associated with it.

ValidationException is thrown if the argument passed is null or it is not a valid routed port InstanceNameId.

Parameters
opContext—Operational context
routedInfId—InstanceNameId of a routed interface

Return Value
List of all the RoutedSubNetworkInterface objects associated for a routed ports
Following associations will be available for each interface:
- NetworkInterfaceName
- NetworkInterfaceStatus
- VLAN
getSviServiceStateForNetworkElements

Returns a collection of Boolean, representing the SVI service state (Enabled / Disabled) for the given device IDs. SVI service state will be null for Catalyst 6500 series switches devices.

ValidationException is thrown if any of the following situation occurs:
- If the argument passed is null.
- If the neInstanceNameIdCol collection is empty or the collection contains object which are not a valid network element InstanceNameId

Parameters
opContext—Operational context
neInstanceNameIds—InstanceNameId corresponding to the network elements

Return Value
List of Boolean representing the SVI service state for the given list of devices.

getSwicthedPortsCountInModule

Returns number of switched ports present in the device for a given module

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId corresponding to the network element
moduleNum—Module Number

Return Value
Number of ports

getSwitchedPortsInModule

Returns all the switched ports present in the device for a given slot. Given the instance name ID of a network element and the slot number of a module, returns a collection of switched interface objects corresponding to the specified module.

ValidationException is thrown if the argument passed is null or it is not a network element InstanceNameId.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId corresponding to the network elements
slotNo—Slot number of the module whose ports are required
Return Value
Set of SwitchedNetworkInterface instances present in a module
Following associations will be available:
- NetworkInterfaceName
- NetworkInterfaceStatus
- PortSetting
- PortCapability
- PortStatus

getSwitchedPortsInModuleByMode

Returns specific switched ports present in the module for a given slot. Given the instance name ID of a network element, slot number of a module and the switchport mode, returns a collection of switched interface objects corresponding to the specified module and mode.

ValidationException is thrown if the argument passed is null or it is not a network element InstanceNameId.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId corresponding to the network elements
slotNo—slot number of the module whose ports are required
mode—switch port mode. If mode is null, return switch ports of all the modes present in a module

Return Value
List of SwitchedNetworkInterface instances present in a module Following associations will be available:
Following associations will be available:
- NetworkInterfaceName
- NetworkInterfaceStatus
- PortSetting
- PortCapability
- PortStatus

getSwitchedPortsInNetworkElement

Returns specific switched ports present in the module for a given mode. Given the instance name ID of a network element and the switchport mode, returns a collection of switched interface objects with specified mode.

ValidationException is thrown if the argument passed is null or it is not a network element InstanceNameId.
getSwitchedPortsWithSecurityAssociations

Return a switched network interface object populated with all security associations.
ValidationException is thrown if the argument passed is null or it is not a valid switched network interface InstanceNameId.

Parameters
opContext—Operational context
interfaceIdCol—InstanceNameId corresponding to the SwitchedNetworkInterface

Return Value
Instance of SwitchedNetworkInterface with security associations populated.
Following associations will be populated for each interface:
- CtsNetworkInterfaceSetting
- Dot1xNetworkInterfaceSetting
- IpSourceGuardSetting
- TrafficStormControlSetting
- Port Security setting. VLAN details will not be populated for this association.

getTotalPortsCountInNetworkElement

Returns number of physical ports present in the device
ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
getTotalRoutedPortsCountInNetworkElement

Returns number of routed ports present in the device.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element

Return Value
Number of ports

getTotalSwitchedPortsCountInNetworkElement

Returns number of switched ports present in the device for a given switch port mode.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element
- mode—switched network interface mode

Return Value
Number of ports

getTunnelInterfacesInNetworkElement

Returns all the tunnel interfaces present in a network element. Given the instance name ID of a network element, returns a collection of tunnel interface objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId of network element
getUdIdServiceStateForNetworkElements

Returns a collection of Boolean, representing the UDLD service state (Enabled / Disabled) for the given device Ids. UDLD service state will be null for Catalyst 6500 series switches devices.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null.
- If the neInstanceNameIdCol collection is empty or the collection contains object which are not a valid network element InstanceNameId

Parameters

- opContext—Operational context
- neInstanceNameIds—InstanceNameId corresponding to the network elements

Return Value

List of Boolean representing the UDLD service state for the given list of devices.

getVlanNetworkInterfacesInNetworkElement

Returns all the VLAN network interfaces (SVIs) present in a network element. Given the instance name ID of a network element, returns a collection of VLAN interface objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of network element

Return Value

List of all the VlanNetworkInterface present in the network element

Following associations will be available for each interface:

- NetworkInterfaceName
- NetworkInterfaceStatus

modifyErrorDetectionSettingInNetworkElement

Modify the existing error detection settings for a given network element Id.
modifyErrorRecoverySettingInNetworkElement

Modify the existing error recovery settings for a given network element Id.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the modifiedErrorRecovery is null.

PropertiesException is thrown if any of the following situation occurs:

- If the recoveryInterval, attribute is invalid.

Example:

- recoveryInterval is out of range. Valid range is between 30 to 86400. Any value configured which
  is less than 30 or greater than 86400 will cause an PropertiesException

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId corresponding to the network element
- modifiedErrorRecovery—modified PortErrorDisableRecoverySetting

Return Value
void

modifyNetworkInterfaceSetting

Modify the network interface global setting for a given network element Id.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the modifiedNetInt is null.
modifyNetworkInterfaces

Modify one or more physical and logical interface (except port-channel) objects. Incase the interface is a kind of IpNetworkInterface then the secondary and helper IP address collection will be updated. If the interface is a mode change (say switched to routed or visa-versa), need to set the AbstractNetworkElement reference.

Following associations will be updated:
- Port Setting

Remaining associations will be take from the existing object in the server in case of modification. If the change is for a port mode (say switched to routed or visa-versa) as well, then what every associations are applicable for both the modes will be moved to new object based on the device platform.

ValidationException is thrown if any of the following situation occurs:
- If the modifiedNetworkInterfaceCol is null or the collection is empty.
- If the modifiedNetworkInterfaceCol contains one or more null element, or the collection contains objects that are not of type NetworkInterface.
- If the modifiedNetworkInterfaceCol contains one or more NetworkInterface objects without AbstractNetworkElement association.
- If the modifiedNetworkInterfaceCol contains objects that are of type SwitchedEtherChannelNetworkInterface or RoutedEtherChannelNetworkInterface. Port-Channels are not supported in this API.
- If the configured IP address (Ipv4AddressPrefix) overlaps with primary or secondary IP address of any other Layer 3 interfaces existing in the device.
- The following ValidationException situations occurs incase modified interface being RoutedSubNetworkInterface
  - If IP address (Ipv4AddressPrefix) being configured to the interface without creating a encapsulation VLAN (VLAN) association.
  - If the configured encapsulation VLAN (VLAN) is not available or already in use by any other interfaces.

PropertiesException is thrown if any of the following situation occurs:
- In the jumboMtu, attribute is invalid.

Example:
- jumboMtu is out of range. Valid range is between 1500 to 9219. Any value configured which is less than 1500 or greater than 9216 will cause an PropertiesException

Parameters

opContext—Operational context

neInstanceId—InstanceNameId of the network element

modifiedNetInt—

Return Value

void
modifyPortModeToRouted

Modify the port mode from switched to routed for a given network interface. On the server, when this method is called all the Layer2 associations will be cleared/retained base on the device platform. Remaining associations will be re-associated to the new object. All attributes present for the old object will be reassigned to the new object.

ValidationException is thrown if any of the following situation occurs:

- If switchedInterfaceNameIdCol collection is null or it is empty.
- If switchedInterfaceNameIdCol collection contains an object that is not of type SwitchedNetworkInterface InstanceName id.
- If switchedInterfaceNameIdCol collection contains a SwitchedNetworkInterface that does not exist in the database.

Parameters

- opContext—Operational context
- switchedInterfaceNameIdCol—List of instance name Ids of switched ports

Return Value

List of newly create routed ports

modifyPortModeToSwitched

Modify the port mode from routed to switched for a given network interface. On the server, when this method is called all the Layer 3 associations will be cleared/retained base on the device platform. Remaining associations will be re-associated to the new object. All attributes present for the old object will be reassigned to the new object.

ValidationException is thrown if any of the following situation occurs:

- If encapsulation type (SubInterfaceEncapsulationType) is configured without specifying encapsulation VLAN.

PropertiesException is thrown if any of the following situation occurs:

- In the modifiedNetworkInterfaceCol collection, if any of the NetworkInterface attribute is not valid.

Example:

- Configuring attribute description with a String of length more than 240 will cause a PropertiesException, because description of an interface can only be up to a maximum of 240 characters.

Parameters

- opContext—Operational context
- modifiedNetworkInterfaceCol—List of instance name Id of create/modified network interfaces
modifyRoutedSubNetworkInterfaces

Modify one or more existing routed sub-interface objects.

Following associations will be updated:

- Port Setting
- Encapsulation VLAN

ValidationException is thrown if any of the following situation occurs:

- If the modifiedRoutedSubNetworkInfs is null or the collection is empty.
- If the modifiedRoutedSubNetworkInfs contains one or more null element, or the collection contains objects that are not of type RoutedSubNetworkInterface.
- If IP address (Ipv4AddressPrefix) being configured to the interface without creating a encapsulation VLAN (VLAN) association
- If the configured IP address (Ipv4AddressPrefix) overlaps with primary or secondary IP address of any other Layer 3 interfaces existing in the device.
- If the configured encapsulation VLAN (VLAN) is not available or already in use by any other interfaces.
- If encapsulation type (SubInterfaceEncapsulationType) is configured without specifying encapsulation VLAN.

PropertiesException is thrown if any of the following situation occurs:

- In the subInfs collection, if any of the RoutedSubNetworkInterface attribute is not valid.

Example:

- Configuring attribute description with a String of length more than 240 will cause an PropertiesException, because description of an interface can only be up to a maximum of 240 characters.
modifySwitchPortMode

Modify the switch port mode for a given list of switch ports. This will enable to specify the new switch port mode (e.g., access, trunk, etc.) and trunk mode.
This API will not re-create a new instance name id. It will just apply the new attributes specified to the existing objects.
Trunk mode can be specified if the port mode is getting changed to Trunk mode. If the trunk mode is not specified, default trunk mode will be change to static mode. Trunk mode will be ignore if the port mode is getting changes to any mode other than trunk.
ValidationException is thrown if any of the following situation occurs:

- If switchedInterfaceNameIdCol collection is null or it is empty.
- If switchedInterfaceNameIdCol collection contains an object that is not of type SwitchedNetworkInterface InstanceId.
- If switchedInterfaceNameIdCol collection contains a SwitchedNetworkInterface that does not exist in the database.

Parameters
opContext—Operational context
switchedInterfaceNameIdCol—List of instance name Ids of routed ports
switchPortMode—New switchport mode
trunkMode—Trunk port mode. This is applicable only if the mode is changed to trunk mode. Based on this mode set, trunk encapsulation and non-negotiate will be set to corresponding defaults. If the trunk mode is static, change the encapsulation to dot1q (if supported by the interface) and non-negotiation to true. Similarly if the trunk mode is changes to dynamic auto/desirable, non-negotiation will be set to false and encapsulation will be set to negotiate (if supported by the interface).

Return Value
void
CHAPTER 17

ImageInstallApp Service

This chapter describes the DCNM web services’ API methods for the ImageInstallApp service.

Information About ImageInstallApp Service

The ImageManagement component allows you to upgrade the devices with the latest images with nondisruptive (dual supervisor devices) and disruptive (single supervisor devices) modes. The ImageInstallJob may contain more than one ImageInstallTask. One ImageInstallTask corresponds to one device within a job. If the device is deleted from the server, all of its corresponding tasks get deleted. ImageInstallApp interface defines the various APIs that handle the image upgrade process.

abortJob

This API is used to abort(delete) the jobs which are scheduled for image upgradation before execution of the job.

ValidationException is thrown if any of the following situation occurs:

- If jobIdList collection is null or it is empty.
- If jobIdList collection contains an element that is not of type ImageInstallJob InstanceNameId.
- If jobIdList collection contains a ImageInstallJob that does not exist in the database.

ImageInstallException is thrown if any of the following situation occurs:

- If the user is trying to abort the currently running ImageInstallJob object

Parameters

- opContext—Operational context
- jobIdList—jobIdList is the list of jobs info which the user is going to abort.

Return Value

void
abortTask

This API is used to abort the selected installation task from the job. (One job may contain more than one tasks.)

ValidationException is thrown if any of the following situation occurs:

- If installTaskCol collection is null or it is empty.
- If installTaskCol collection contains an element that is not of type ImageInstallTask InstanceNameId.
- If installTaskCol collection contains an ImageInstallTask that does not exist in the database.

Parameters

opContext—Operational context
installTaskCol—taskcol is list of tasks which the user is going to abort from the job.

Return Value

void

createJob

This API is used to schedule a new job for image upgradation. Creates jobInstance. Job cannot be created without ImageInstallTask association. One ImageInstallJob may contain more than one ImageInstallTask. One ImageInstallTask corresponds to one device within a job.

Following associations are to be provided for the given ImageInstallJob:

1. ImageInstallTaskReference corresponding to the specified ImageInstallJob. ImageInstallTask reference
2. AbstractNetworkElementReference corresponding to the specified ImageInstallTask
   AbstractNetworkElement reference

ValidationException is thrown if any of the following situation occurs:

- If newJobInfoCol is null or empty.
- If elements in newJobInfoCol is not a valid type of ImageInstallJob or with ImageInstallTask reference not specified.

IntegrityException is thrown if any of the following situation occurs:

- If a Job with same jobId already exist.

Parameters

opContext—Operational context
newJobInfoCol—List of ImageInstallJob objects to be created.

Return Value

List of InstanceNameId corresponding to the newly created ImageInstallJob.
**deleteJob**

This API is used to delete the scheduled job. Deletes the job objects corresponding to the given list of InstanceNameId. The corresponding ImageInstallJob and its associations will be deleted.

ValidationException is thrown if any of the following situation occurs:

- If jobIdList collection is null or it is empty.
- If jobIdList collection contains an element that is not of type ImageInstallJob InstanceNameId.
- If jobIdList collection contains a ImageInstallJob that does not exist in the database.
- If the user is trying to delete the currently running job

ImageInstallException is thrown if any of the following situation occurs:

- If the user is trying to delete the currently running ImageInstallJob object

**Parameters**

- opContext—Operational context
- jobIdList—is the List of jobs that needs to be deleted.

**Return Value**

void

**fetchDiskFreeSpace**

This API is used to know the free available disk space in bootflash.

**Parameters**

- opContext—Operational context
- neInstanceNameId—neInstanceNameId is the InstanceNameId for a given network element.

**Return Value**

List free available disk space.

**fetchDiskFreeSpaceFromStandbySup**

This API is used to know the free available disk space in bootflash for standby.

**Parameters**

- opContext—operational context
- neInstanceNameId—neInstanceNameId is the InstanceNameId for a given network element.

**Return Value**

List free available disk space for the given devices.
### getAllInstallTasks

This API is used to fetch all the ImageInstallTask objects for the given Network Element. If null is passed for network Element InstanceNameId return all the list of ImageInstallTask objects. 

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is not InstanceNameId of AbstractNetworkElement.
- If the corresponding object for neInstanceNameId does not exist.

**Parameters**

- opContext—Operational context

**Return Value**

List of installtasks.

### getAllJobs

This API is used to get the ImageInstallJob objects corresponding to a status passed as a parameter. If the status passed is null then the all ImageInstalljob objects will be returned irrespective of the status.

**Parameters**

- opContext—Operational context
- aInstallationStatus—aInstallationStatus is the status which the user needs the list of jobInfo

aInstallationStatus might be any of the following:

1) InstallationStatus.COMPLETED return all the list of completed jobs.
2) InstallationStatus.RUNNING return all the list of active jobs.
3) InstallationStatus.SCHEDULED return all the list of pending jobs.

**Return Value**

List of all the installJobs of specified status type.

### getFailureLog

Returns the failure log for a job collection.

**Parameters**

- opContext—Operational context
- jobCol—List of InstanceNameId for jobs.

**Return Value**

List of failure logs.

### getJob

This API is used to get the specific job and corresponding task details.
**getstatusofjobinnetworkelement**

This API is used to get last completed job or current job status for a given NetworkElement. If no current or last completed jobs found in the given network element, API returns null.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null or not an InstanceNameId of AbstractNetworkElement.

**Parameters**

- **opContext**—Operational context
- **neInstanceNameId**—neInstanceNameId is the InstanceNameId for a given network element.
- **aInstallationStatus**—If null is passed for aInstallationStatus, return last completed job status for a given network element. If InstallationStatus.Running is passed for aInstallationStatus, return the current job status for a given network element.

**Return Value**

InstallationStatus, last completed job or current job status for a given network element.

**modifyjob**

This API is used to modifies the scheduled job with modified job info. Modify attributes for ImageInstallJob instances. User cannot modify the jobId. User can update the scheduleTime and also can update the ImageInstallTask association. That is user can add or delete the ImageInstallTask from the given ImageInstallJob instance.

ValidationException is thrown if any of the following situation occurs:

- If the modifiedJobInfo is null
- If the corresponding ImageInstallJob object does not have equivalent persisted ImageInstallJob object.

PropertiesException is thrown if any of the following situation occurs:

- If any of the ImageInstallJob attribute is not valid.

ImageInstallException is thrown if any of the following situation occurs:

- If the user is trying to modify the currently running ImageInstallJob object

**Parameters**

- **opContext**—Operational context
- **modifiedJobInfo**—modifiedJobInfoCol is the modifiedInfo of the scheduled jobs.
**performVersionComaptability**

This API is used to get the version compatibility log info between kickstart and system images. For example if the specified kickstart and system images not compatible(belonging to different versions) API return log info contain the log messages will be as following. "Install has failed. Return code 0x40930024 (Images specified are not upgradeable). Please identify the cause of the failure, and try 'install all' again."

ValidationException is thrown if any of the following situation occurs:

- If neIdCol collection is null or it is empty.
- If neIdCol collection contains an element that is not of type AbstractNetworkElement InstanceNameId.
- If taskCol collection is null or it is empty.
- If taskCol collection contains an element that is not of type ImageInstallTask InstanceNameId.
- If jobCol collection is null or it is empty.
- If jobCol collection contains an element that is not of type ImageInstallJob InstanceNameId.

**Parameters**

- `opContext`—Operational context
- `neIdCol`—InstanceId of the network elements
- `taskCol`—is the collection of ImageInstallTask objects
- `jobCol`—is the collection of ImageInstallJob objects

**Return Value**

List is the log information, by specifying whether both the kickstart and system images are compatible or not.

**resumeJob**

This API is used to resume the job which is suspended form the execution

ValidationException is thrown if any of the following situation occurs:

- If jobIdList collection is null or it is empty.
- If jobIdList collection contains an element that is not of type ImageInstallJob InstanceNameId.
- If jobIdList collection contains a ImageInstallJob that does not exist in the database.

ImageInstallException is thrown if any of the following situation occurs:

- If the user is trying to resume the unsuspended ImageInstallJob object

**Parameters**

- `opContext`—Operational context
suspendJob

This API is used to suspend(stop) the jobs from the execution and may resume some time later. ValidationException is thrown if any of the following situation occurs:

- If jobIdList collection is null or it is empty.
- If jobIdList collection contains an element that is not of type ImageInstallJob InstanceNameId.
- If jobIdList collection contains a ImageInstallJob that does not exist in the database.

ImageInstallException is thrown if any of the following situation occurs:

- If the user is trying to suspend the currently running ImageInstallJob object.

Parameters

- opContext—Operational context
- jobIdList—jobIdList is jobs info which the user is going to suspend.

Return Value

void
IpSourceGuardApp Service

This chapter describes the DCNM web services’ API methods for the IpSourceGuardApp service.

Information About IpSourceGuardApp Service

This chapter defines the APIs exposed by the IP Source Guard service feature.

addIpSourceBindings

Creates a static binding entry, given the binding entry object.
ValidationException is thrown if any of the following situation occurs:
• If ipSourceBinding object does not contain value for abstractNetworkElementRef.
• If the AbstractNetworkElement specified by abstractNetworkElementRef in the ipSourceBinding object does not exist in the database.
• If the ipSourceBinding object does not contain value for networkInterfaceRef.
• If the NetworkInterface specified by networkInterfaceRef in the ipSourceBinding object does not exist in the database.
• If the ipSourceBinding object does not contain value for vlanRef.
• If the (@link com.cisco.dcbu.dcm.model.VLAN.VlanExternal} specified by vlanRef in the ipSourceBinding object does not exist in the database.

IntegrityException is thrown if the ipSourceBinding object already exist.
PropertiesException is thrown if any of the following situation occurs:
• If any of the attributes of the ipSourceBinding object is not valid.
• If the lease expiry time specified for the ipSourceBinding object, got expired.

Parameters
ipSourceBindings—A collection of an object of type IpSourceBinding. This object represents an entry in IP source binding table in the device.

Return Value
A collection of the new binding entry.
deleteAllIpSourceBindingsInInterfaces

Deletes all static binding entries configured in an interface, given the InstanceNameId of the interface. ValidationException is thrown if any of the following situation occurs:

- If the argument passed interfaceInstanceNameId is not a valid interface instance name ID.
- If the interface does not contain any static IP source binding associated to it.

Parameters

interfaceInstanceNameIds— A collection of InstanceNameId of the interface, whose static binding entries are to be deleted.

Return Value

void

deleteAllIpSourceBindingsInNetworkElements

Deletes all static binding entries in a given network element, given the InstanceNameId of the network element. ValidationException is thrown if any of the following situation occurs:

- If the argument passed neInstanceNameId is not a valid network element instance name ID.
- If the network element does not exist in the database.
- If the network element does not contain any IpSourceBinding associated to it.

Parameters

networkElementIds— A Collection InstanceNameId of the network element in which bindings are to be deleted.

Return Value

void

deleteAllIpSourceBindingsInVlanOfAnInterface

Deletes all static binding entries configured in a given VLAN in an interface, given the InstanceNameId of the interface and VLAN. ValidationException is thrown if any of the following situation occurs:

- If the argument interfaceInstanceNameId is null or not a valid interface instance name ID.
- If the interface does not exist in the database.
- If the argument vlanInstanceNameId is null or not a valid VLAN instance name ID.
- If the VLAN does not exist in the database.
- If the VLAN does not contain any IpSourceBinding associated to it.
Chapter 18      IpSourceGuardApp Service

Send document comments to nexus7k-docfeedback@cisco.com

deleteIpSourceBindings

Parameters
opContext—Operational context
interfaceInstanceNameId—InstanceNameId of the interface in which bindings are to be deleted for the given VLAN.
vlanInstanceNameId—InstanceNameId of the VLAN in which the bindings are to be deleted.

Return Value
void

deleteIpSourceBindings

Deletes a static binding entry from the IP source binding table. InstanceNameId of the IP source binding.
ValidationException is thrown if any of the following situation occurs:
• If ipSourceBindingInstanceNameId is null or not a valid IP source binding instance name ID.
• If the IP source binding does not exist in the database.

Parameters
ipSourceBindingInstanceNameIds—A Collection of InstanceNameId of the IP source binding that is to be deleted.

Return Value
void

disableIpSourceGuardInNetworkElement

Disables IP source guard in all interfaces in a network element, given the instance name ID of the network element.
ValidationException is thrown if any of the following situation occurs:
• If the argument passed is null or it is not a valid network element InstanceNameId.
• If the network element does not exist in the database.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId of the network element.

Return Value
void

disableIpSourceGuardOnInterfaces

Disables IP source guard in a given collection of interfaces.
ValidationException is thrown if any of the following situation occurs:
enableIpSourceGuardOnInterfaces

Enables IP source guard in a given collection of interfaces.
ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid switched network interface InstanceNameId.
- If the interface does not exist in the database.

**Parameters**
- opContext—Operational context
- interfaceInstanceNameIds—A collection of InstanceNameId of interfaces in which IP source guard is to be disabled.

**Return Value**
void

getInterfacesWithIpSourceBindingsInNetworkElement

Returns the interfaces having static bindings configured in them, given the InstanceNameId of the network element. The network element is the device in which the interface is present.
ValidationException is thrown any of the following situations occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

**Parameters**
- opContext—Operational context
- neInstanceNameId—InstanceNameId of network element

**Return Value**

A collection of interfaces in the network element in which static bindings are configured. Returned collection will have InstanceNameIds of interface objects having static bindings. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- IpSourceBindingassociation.
getIpSourceBindings

Returns IpSourceBinding object corresponding to the instance Id.
ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid IP source binding InstanceNameId.
- If the IP source binding does not exist in the database.

**Parameters**

- `ipSourceBindingInstanceNameIds`— A collection of InstanceNameId of IpSourceBinding object representing an entry in the IP Source binding table.

**Return Value**

A collection of an object of type IpSourceBinding. In the returned object, only the following associations will be present, and all other associations will be cleared.

- AbstractNetworkElement association.
- NetworkInterface association.
- VlanExternal association.

getIpSourceBindingsInInterface

Returns the static bindings configured in an interface, given the InstanceNameId of the interface.
ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.

**Parameters**

- `opContext`—Operational context
- `interfaceInstanceNameId`— InstanceNameId of the interface

**Return Value**

A collection of static binding configured in the given interface. Returned collection will have objects of type IpSourceBinding. In the returned list of objects, only the following associations will be present and other associations will be cleared.

- AbstractNetworkElement association.
- NetworkInterface association.
- VlanExternal association.
getIpSourceBindingsInVlanOfAnInterface

Returns the static bindings configured in a VLAN in an interface, given the InstanceNameId of the VLAN and the interface.

ValidationException is thrown if any of the following situation occurs:
- If the argument interfaceInstanceNameId passed is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.
- If the argument vlanInstanceNameId passed is null or it is not a valid VLAN InstanceNameId.
- If the VLAN does not exist in the database.

Parameters
- opContext—Operational context
- interfaceInstanceNameId—InstanceNameId of the interface
- vlanInstanceNameId—InstanceNameId of a VLAN in the interface

Return Value
A collection of static binding configured in given VLAN in an interface. Returned collection will have objects of type IpSourceBinding. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- AbstractNetworkElement association.
- NetworkInterface association.
- VlanExternal association.

getIpSourceGuardDisabledInterfacesInNetworkElement

Returns all the interfaces in which IP Source Guard is disabled in a given network element.

ValidationException is thrown any of the following situations occurs:
- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId of the network element.

Return Value
A collection of SwitchedNetworkInterface in which IP source guard is disabled, in the given network element. In the returned list of objects, only the following associations will be present and other associations will be cleared.
- IP source guard association.
- NetworkInterfaceName association.
**getIpSourceGuardEnabledInterfacesInNetworkElement**

Returns all the IP source guard enabled interfaces in a given network element. ValidationException is thrown if any of the following situations occurs:

- If the argument passed is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

**Parameters**

- `opContext`—Operational context
- `neInstanceNameId`—InstanceNameId of the network element.

**Return Value**

A collection of IP source guard enabled SwitchedNetworkInterface in the given network element. In the returned list of objects, only the following associations will be present and other associations will be cleared:

- IP source guard association.
- NetworkInterfaceName association.

**getIpSourceGuardSettingOnInterfaces**

Returns the IP source guard setting corresponding to a given interface. ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.

**Parameters**

- `opContext`—Operational context
- `interfaceInstanceNameIds`—A collection of InstanceNameId of interfaces.

**Return Value**

A collection of IpSourceGuardSetting object corresponding to interfaces.

**getNumberOfDynamicBindingsInVlan**

Returns the number of dynamic bindings configured in a VLAN, given the InstanceNameId of the VLAN. ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid VLAN InstanceNameId.
- If the VLAN does not exist in the database.

**Parameters**

- `opContext`—Operational context
getNumberOfIpSourceBindingsInInterfaces

Returns the number of static bindings configured in a collection of Interfaces, given the InstanceNameId of the Interfaces.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid Interface InstanceNameId.

Parameters

interfaceInstanceNameIds— A collection of InstanceNameId of Interfaces. *

Return Value

A collection of integer values representing the number of static bindings configured in Interfaces.

getNumberOfIpSourceBindingsInVlans

Returns the number of static bindings configured in a collection of VLANs, given the InstanceNameId of the VLANs.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid VLAN InstanceNameId.
- If the VLAN does not exist in the database.

Parameters

opContext—Operational context
vlanInstanceNameIds— A collection of InstanceNameId of VLANs.

Return Value

A collection of integer values representing the number of static bindings configured in VLANs

modifyIpSourceBindings

Updates the server with the modified Static IP Source binding entry.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed ipSourceBinding is null.
- If the original ipSourceBinding does not exist in the database.

PropertiesException is thrown if any of the attributes of IpSourceBinding is not valid.

Example:
modifyIpSourceBindings

- If the value for IpAddress is not specified.
- If the value for the MacAddress is not specified
- If the value for the lease expiry time is not specified and etc.

Parameters

ipSourceBinding—— A collection of modified IP Source binding entry of type IpSourceBinding

Return Value

void
InventoryApp Service

This chapter describes the DCNM web services’ API methods for the InventoryApp service.

Information About InventoryApp Service

This chapter defines the APIs exposed by the Inventory service feature.

getAllChassisInNetwork

Returns a collection of all the Chassis managed by the application. Return an empty collection when no chassis is managed by the application.

Parameters
opContext—Operational context.

Return Value
List of Chassis object.

Following associations will be available for each Chassis object:

- NetworkElement.
- Slot collection.
- RedundancySetting.
- RedundancyStatus.
- ChassisPowerSupplySetting.

getAllNetworkElements

Returns all the Devices managed by the application. Returns empty collection when no devices are managed by application.
getAllPhysicalLinkInNetwork

Returns a collection of physical links present in a network element. Given InstanceNameId of the network element, returns a collection of physical link objects.

ValidationException is thrown if any of the following situation occurs:
- If the argument passed is null or it is not a valid network element InstanceNameId
- If a valid non-existing network element InstanceNameId is passed.

Parameters
- opContext—Operational context.
- neInstanceNameId—InstanceNameId of the network element.

Return Value
- List of PhysicalLink objects corresponding to NetworkElement.

following association will be available for each PhysicalLink objects:
- PhysicalPort collection.
- NetworkInterface.

getAllPhysicalLinks

Returns list of Physical Links present in the network.

Parameters
- opContext—Operational context.

Return Value
- List of PhysicalLink objects.

getCardsInChassis

Returns a collection of cards or modules available in the given chassis. The returned collection of card objects can be any/all of the following card objects:
- ProcessorCard.
**getCdpNeighbors**

Returns a collection of network elements that physically connected to a network element. Given a InstanceNameId of the network element, returns a list of network element objects.

ValidationException is thrown if any of the following situation occurs:

- If srcInstanceNameId is null or if it is not a valid network element InstanceNameId

**Parameters**

- opContext—Operational context.
- srcInstanceNameId—InstanceNameId of a NetworkElement.

**Return Value**

List of NetworkElement objects that has a physical link with the input NetworkElement.

Following association will be available for each NetworkElement object:

- Chassis.

**getChassis**

Returns a collection of chassis objects managed by the application. Given the list of chassis InstanceNameId returns collection of Chassis objects.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null
fetchCurrentSensorsInCard

Given the InstanceNameId of a Card, returns the Sensors that are associated to it. The Sensors will contain details about current temperature, major threshold, minor threshold and sensor status. ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null.
- if Card InstanceNameId is invalid.
- if a valid non-existing Card InstanceNameId is passed.

Parameters
opContext—Operational context.
cardInstanceNameId—Card InstanceNameId

Return Value
List of Sensors associated with the Card.
Following associations will be available for each Sensor objects:
- Threshold.
- SensorStatus.
- SensorDataType.

fetchCurrentSensorsInChassis

Given a collection of chassis InstanceNameId, returns the sensors associated to it. The Sensors will contain details about current temperature, major threshold, minor threshold and sensor status. ValidationException is thrown if any of the following situation occurs:

- If the chassis InstanceNameId collection is empty or the collection contains invalid chassis InstanceNameId

Parameters
opContext—Operational context.
chassisIds—list of InstanceNameId of the chassis.

Return Value
List of Chassis object.
Following associations will be available for each Chassis object:
- NetworkElement.
- Slot collection.
- RedundancySetting.
- ChassisPowerSupplySetting.
getFansInChassis

Returns a collection of all fan present in the chassis. Given the InstanceNameId of the chassis, returns a collection of fan objects. When the chassis does not contain any fan units, this API will return a empty collection.

ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null
- if a valid non-existing chassis InstanceNameId is passed.

Parameters
opContext—Operational context.
chassisId—InstanceNameId of the chassis.

Return Value
List of Fan objects present in the chassis.
Following associations will be available for each Fan objects:
- Sensor collection.
- Chassis.
- FanStatus.

getFileSystemsInChassis

Deprecated.

Returns a collection of all file system present in the chassis. Given the InstanceNameId of the chassis, returns a collection of file system objects. When the chassis does not contain any file systems, this API will return a empty collection.

ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null or it is not a valid chassis InstanceNameId

Parameters
opContext—Operational context.
chassisInstanceNameId—List of Chassis InstanceNameId

Return Value
List of Sensors associated with the Chassis.
Following associations will be available for each Sensor objects:
- Threshold.
- SensorStatus.
- SensorDataType.
getNetworkCardsInChassis

Returns a collection of all network cards present in the chassis. Given the InstanceNameId of the chassis, returns a collection of network card objects. Empty collection is returned when the chassis does not contain any network cards.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid chassis InstanceNameId
- If a valid non-existing chassis InstanceNameId is passed.

Parameters

- opContext—Operational context.
- chassisId—InstanceNameId of the chassis.

Return Value

List of NetworkCard objects present in the chassis.

Following associations will be available for each NetworkCard objects:

- Slot.
- Card sub-card collection for that NetworkCard if present.
- AsicChip collection for that NetworkCard if present.
- NetworkCardEtherChannelLoadBalanceSetting.

getPhysicalLinks

Returns a collection of physical links that exist between two network elements. Given the InstanceNameId of two network elements, this API will return a collection of physical link objects.

ValidationException is thrown if any of the following situation occurs:

- If both srcInstanceNameId, destInstanceNameId are null or is not a valid InstanceNameId of the network element.
Send document comments to nexus7k-docfeedback@cisco.com

getPhysicalPorts

Returns a collection of physical ports present in a particular slot of the chassis. Given theInstanceNameId of the chassis and the slot number, returns a collection of physical port objects. ValidationException is thrown if any of the following situation occurs:

- If the passed chassis InstanceNameId is null or it is not a valid chassis InstanceNameId.
- If a valid non-existing chassis InstanceNameId is passed
- If the slot number passed is null.

Parameters

opContext—Operational context.
aChassisInstanceNameId—InstanceNameId of the chassis.
slotNumber—An Integer slot number.

Return Value

List of PhysicalPort objects corresponding to Chassis and the slotNumber.
Following associations will be available for each PhysicalPort objects:

- NetworkInterface
- Card

getPhysicalPortsInSlot

Returns a collection of physical ports present in a particular slot of the chassis. Given the InstanceNameId of the chassis and the slot number, returns a collection of physical port objects. ValidationException is thrown if any of the following situation occurs:

- If the passed chassis InstanceNameId is null or it is not a valid chassis InstanceNameId.
- If a valid non-existing chassis InstanceNameId is passed
- If the slot number passed is null.

Parameters

opContext—Operational context.
**getPowerSuppliesInChassis**

Returns a collection of all power supplies present in the chassis. Given the InstanceNameId of the chassis, returns a collection of power supply objects. When the chassis does not contain any power supply, this API will return an empty collection.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid chassis InstanceNameId
- If a valid non-existing chassis InstanceNameId is passed.

**Parameters**

- opContext—Operational context.
- chassisId—InstanceNameId of the chassis.

**Return Value**

List of PowerSupply objects present in the chassis.

Following associations will be available for each PowerSupply objects:

- Sensor collection.
- Chassis.
- PowerSupplyStatus collection.

**getProcessorCardsInChassis**

Returns a collection of all processor cards present in the chassis. Given the InstanceNameId of the chassis, returns a collection of processor card objects. When the chassis does not contain any processor cards, this API will return an empty collection.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid chassis InstanceNameId
- If a valid non-existing chassis InstanceNameId is passed.

**Parameters**

- opContext—Operational context.
- chassisId—InstanceNameId of the chassis.
getSensorsInChassis

Returns a collection of all sensors present in the chassis. Given the InstanceNameId of the chassis, returns a collection of sensor objects. When the chassis does not contain any sensors, this API will return an empty collection.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid chassis InstanceNameId
- If a valid non-existing chassis InstanceNameId is passed.

Parameters

opContext—Operational context.
chassisId—InstanceNameId of the chassis.

Return Value

List of Sensor objects corresponding to Chassis.

Following associations will be available for each Sensor objects:

- Threshold.
- SensorStatus.
- SensorDataType.

getServiceCardsInChassis

Returns a collection of all service cards present in the chassis. Given the InstanceNameId of the chassis, returns a collection of service card objects. When the chassis does not contain any service cards, this API will return an empty collection.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid chassis InstanceNameId
- If a valid non-existing chassis InstanceNameId is passed.

Parameters

opContext—Operational context.
chassisId—InstanceNameId of chassis.
getSlotsInChassis

Returns a collection of slots available in the given chassis. When the chassis does not contain any slots, this API will return an empty collection.

ValidationException is thrown if any of the following situation occurs:
- If InstanceNameId of chassis is null or if it is not a valid chassis InstanceNameId

Parameters
opContext—Operational context.
chassisInstanceNameId—InstanceNameId of a Chassis.

Return Value
List of Slot objects corresponding to the input chassis InstanceNameId.
Following association will be available for each Slot object:
- Chassis association for each Slot object.

getSlotsInNetworkElement

In case of Catalyst 6500 series switches, slots present in the device will be returned. In case of Nexus 7000 series switch, slots present in the VDC will be returned Returns a collection of slot number of the slots present in Virtual Network Element for Nexus 7000 series switch. Ccollection of slot number of the slots present in Device is returned for Catalyst 6500 series switches.

Parameters
opContext—Operational context.
aneInstanceNameId—InstanceNameId of AbstractNetworkElement.

Return Value
List of SlotNumbers

getSoftwareIdentitiesInNetwork

Returns a collection of all software identities from the active processor cards present in the network elements. Given a list of InstanceNameId of the network elements, this API returns a collection of software identities collected from the active processor cards.
ValidationException is thrown if any of the following situation occurs:

- If network element InstanceNameId collection is null, empty or if the collection does not contain InstanceNameId of the network element or virtual network element.

**Parameters**

- `opContext`—Operational context.
- `anelInstanceNameIds`—List of InstanceNameId of the NetworkElement.

**Return Value**

List of SoftwareIdentity objects corresponding to NetworkElement or VirtualNetworkElement

g**etSystemUpTime**

Deprecated.

Returns the System uptime of the device for the given chassis InstanceNameIds. System up time returned is in the same order of chassis InstanceNameIds passed.

ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null
- if the chassisIds collection is empty or the collection contains objects which are not a valid chassis InstanceNameIds

**Parameters**

- `opContext`—Operational context.
- `chassisInstanceNameIds`—List of Chassis InstanceNameId

**Return Value**

List containing System up time.
KeyChainApp Service

This chapter describes the DCNM web services’ API methods for the KeyChainApp service.

Information About KeyChainApp Service

Keychain management allows you to configure shared secrets on all the entities that exchange secrets (such as keys before establishing trust with each other). Routing protocols and network management applications often use authentication to enhance security while communicating with peers. The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Modify APIs—Modify existing key chain and key chain entries.
- Delete APIs—Delete existing key chains from the device.

createKeyChains

Creates a keyChain in the associated NetworkElement

Following associations are to be provided for the given KeyChain:

- NetworkElement corresponding to the KeyChain AbstractNetworkElement reference

ValidationException is thrown if any of the following situation occurs:

- If keyChainCol is null or empty or it is not of type KeyChain.

Parameters

opContext—Operational context
keyChainCol—List of KeyChain objects

Return Value

List of InstanceNameIds of (com.cisco.dcbu.dcm.model.auth.KeyChain)

deleteKeyChains

Deletes one or more KeyChains configured in the device given a list of instanceNameIds of KeyChain.
getKeyChains

Returns the keyChain configured in the specified network Elements.
ValidationException is thrown if any of the following situation occurs:
- If keyChainIdsCol is null or it is not of type InstanceNameId.
- If keyChainIdsCol is not a valid KeyChain InstanceNameId.

Parameters
- opContext—Operational context
- keyChainIdsList—List of InstanceNameIds of KeyChain

Return Value
- void

getKeyChainsById

Returns the keyChain corresponding to the instance name id.
ValidationException is thrown if any of the following situation occurs:
- If keyChainInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If keyChainInstanceNameIdCol is not a valid InstanceNameId of KeyChain object.

Parameters
- opContext—Operational context
- keyChainInstanceNameIdCol—List of InstanceNameId if one or more KeyChain.

getKeyChainsByld

Returns the keyChain corresponding to the instance name id.
ValidationException is thrown if any of the following situation occurs:
- If keyChainInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If keyChainInstanceNameIdCol is not a valid InstanceNameId of KeyChain object.

Parameters
- opContext—Operational context
- keyChainInstanceNameIdCol—List of InstanceNameId if one or more KeyChain.
modifyKeyChains

Modifies one or more existing KeyChain objects.
ValidationException is thrown if any of the following situation occurs:
- If keyChainCol is null or empty or it is not of type KeyChain.

Parameters
- opContext—Operational context
- keyChainCol—List of modified KeyChain objects

Return Value
List of modified KeyChain Objects persisted in the database. The returned objects contain the following associations, all other associations will be cleared.
- AbstractNetworkElement for KeyChain
- KeyChainEntry association for KeyChain
- KeyChain association for KeyChainEntry
Chapter 20      KeyChainApp Service

modifyKeyChains

Send document comments to nexus7k-docfeedback@cisco.com
LicensingApp Service

This chapter describes the DCNM web services’ API methods for the LicensingApp service.

Information About LicensingApp Service

This chapter defines the APIs exposed by the Licensing service feature of the DCNM server.

getDetailedLicenseInfo

Returns the detailed license information of DCNM server.
For each valid license file, there will be corresponding LicenseInfo object.

Return Value
The detailed license information

getHostIdentifier

Returns the host identifier of this DCNM server.

Parameters
opContext—Operational context of this API call

Return Value
The host id of this DCNM server

getLicenseInfo

Returns the license information of the server.

Parameters
opContext—Operational context of this API call
performActivateLicenses

Re-reads the license files and refreshes the licenses information.
If there is a change in license information, an event, whose source instance set as modified instance of LicenseInfo, will be sent

Parameters
opContext—Operational context of this API call

Return Value
void
LogApp Service

This chapter describes the DCNM web services’ API methods for the LogApp service.

Information About LogApp Service

This chapter defines the logging services exposed by the DCNM server.

getDefaultLogLevel

Returns the default log level configured in DCNM server.

**Return Value**
The default logging level

getLogLevel

Returns the log level of the given component.
If the returned LogLevel is not null, it means that user has explicitly set the returned log level to the given component. If the returned level is null, the user has not set any log level for this component explicitly and the default log level (returned by getDefaultLogLevel(OpContext)) has to be used for this component.

**Parameters**
component—component whose log level needs to be returned

**Return Value**
The log level of the given component

log

The given message is loged under Info
modifyDefaultLogLevel

Sets default logging level to the given feature/service component.
If the component given is null, the default level will be set for all the DCNM feature components and services.

Parameters
defaultLevel—logging level to be set as default.

Return Value
void

modifyLogLevel

The given component's logging level will be changed to the given logging level.
If the given component is null, the given log level is set to all features whereas if given log level is null, then the given component will be set with default level.

Parameters
component—component whose log level is to be changed
logLevel—new log level need to be set to given component

Return Value
void
NacLpIpApp Service

This chapter describes the DCNM web services’ API methods for the NacLpIpApp service.

Information About NacLpIpApp Service

NacLpIpApp allows a host that is seeking network access to have an up-to-date virus signature set, the most current operating system patches, and to be free from infection. This enforcement, called posture validation, limits damage to the network from viruses, worms, and spyware.

Hosts that pass posture validation will be granted access to the network. Hosts that fail posture validation will be either denied access or provided restricted access that is sufficient for remediation. The remediation server has a repository of updates for antivirus software and security patches. Hosts that fail posture validation are forwarded to this remediation server to enable them to download or upgrade antivirus software and operating system security patches.

NAC APIs are defined with the following categories:
1. Query and Get APIs—Used to query data from the persisted database.
2. Create APIs—Used to create a new Policy, Profile, or ExemptedHost.
3. Modify APIs—Used to modify a Policy, Profile, or ExemptedHost.
4. Delete APIs—Used to delete a Policy or Profile.
5. Bind and Unbind APIs—Used to bind or unbind the association between two features.
6. Add and Remove APIs—Used to add or remove the association between two features.

addExceptionListHostsToIdentityProfile

Adds Exception List hosts to identity policy.

ValidationException is thrown if any of the following situation occurs:
• If exceptionListHostCol collection is null or it is empty.
• If exceptionListHostCol collection contains an element that is not of type IpAdmissionControlRule InstanceNameId.
• If identityProfileId is null or it is empty.
• If identityProfileId contains an element that is not of type IdentityProfile InstanceNameId.
• If identityProfileId contains a IdentityProfile that does not exist in the database.
bindAccessListToIdentityPolicies

Assigns an access list to a collection of identity policies. For an access list to be bound to an identity policy, name of the list is sufficient. The access list need not have been configured in the device. This API addresses this pre-provisioning configuration. Network element InstanceNameId can be obtained from identityPolicyIds.

ValidationException is thrown if any of the following situation occurs:
- If identityPolicyIdCol collection is null or it is empty.
- If identityPolicyIdCol collection contains an element that is not of type IdentityPolicy InstanceNameId.
- If identityPolicyIdCol collection contains a IdentityPolicy that does not exist in the database.
- If aclName is null or it is empty.

Parameters
- opContext—Operational context
- aclName—Name of the IP Access List
- identityPolicyIdCol—a collection of InstanceNameId of identity policies

Return Value
void

bindEapOverUdpValidationToNetworkInterfaces

Applies the given EapOUpdValidation object to a given set of interfaces. *

ValidationException is thrown if any of the following situation occurs:
- If interfaceNameIdCol collection is null or it is empty.
- If interfaceNameIdCol collection contains an element that is not of type IpAdmissionControlRule InstanceNameId.
- If interfaceNameIdCol collection contains a SwitchedNetworkInterface that does not exist in the database.
- If eapOUpdValidation is null or it is empty.

Parameters
- opContext—Operational context
bindIdentityPolicyToExceptionListHosts

Assigns a given identity policy to a given collection of statically configured exception list hosts. ValidationException is thrown if any of the following situation occurs:

- If exceptionListHostIdCol collection is null or it is empty.
- If exceptionListHostIdCol collection contains an element that is not of type ExceptionListHost InstanceNameId.
- If exceptionListHostIdCol collection contains a ExceptionListHost that does not exist in the database.
- If identityPolicyId is null or it is empty.
- If identityPolicyId contains an element that is not of type IdentityPolicy InstanceNameId.
- If identityPolicyId contains a IdentityPolicy that does not exist in the database.

**Parameters**

- opContext—Operational context
- identityPolicyId—InstanceNameId of IdentityPolicy
- exceptionListHostIdCol—a collection of InstanceNameId of ExceptionListHost

**Return Value**

void

bindIdentityPolicyToExceptionListHostsByName

Assigns pre-provisioned identity policy to a collection of exception list hosts. For Identity policy to be bound to a collection of exception list hosts, identity policy need not have been configured in the device. This API addresses this pre-provisioning configuration.

ValidationException is thrown if any of the following situation occurs:

- If exceptionListHostIdCol collection is null or it is empty.
- If exceptionListHostIdCol collection contains an element that is not of type ExceptionListHost InstanceNameId.
- If exceptionListHostIdCol collection contains a ExceptionListHost that does not exist in the database.
- If policyName is null or it is empty.
**bindIpAdmissionControlRulesToNetworkInterfaces**

Applies an IP admission control rule on a collection of Switched Network interfaces.

ValidationException is thrown if any of the following situation occurs:

- If `interfaceNameIdCol` collection is null or it is empty.
- If `interfaceNameIdCol` collection contains an element that is not of type `IpAdmissionControlRuleInstanceNameId`.
- If `interfaceNameIdCol` collection contains a `IpAdmissionControlRule` that does not exist in the database.
- If `ipAdmissionControlRuleId` is null or it is empty.
- If `ipAdmissionControlRuleId` contains an element that is not of type `IpAdmissionControlRuleInstanceNameId`.
- If `ipAdmissionControlRuleId` contains a `IpAdmissionControlRule` that does not exist in the database.

**Parameters**

- `opContext`—Operational context
- `ipAdmissionControlRuleId`—InstanceNameId of the `IpAdmissionControlRule` to be applied on a set of interfaces
- `interfaceNameIdCol`—a collection of `InstanceNameId` of interfaces on which the given IP admission control rule has to be applied.

**Return Value**

void

**clearIpAdmissionControlRuleFromInterfaces**

Clears the given IP admission control rule applied on a collection of interfaces.

ValidationException is thrown if any of the following situation occurs:

- If `interfaceNameIdCol` is null or it is empty.
- If `interfaceNameIdCol` contains invalid Switched Network Interface `InstanceNameId` or null value.
- If there is no equivalent Switched Network Interfaces object with the given `InstanceNameId` in the `interfaceNameIdCol`.

**Parameters**

- `opContext`—Operational context
- `ipAdmissionControlRuleId`—InstanceNameId of the `IpAdmissionControlRule` to be applied on a set of interfaces
- `interfaceNameIdCol`—a collection of `InstanceNameId` of interfaces on which the given IP admission control rule has to be applied.

**Return Value**

void
createIdentityPoliciesInNetworkElement

Creates a collection of identity policies in a network element.

ValidationException is thrown if any of the following situation occurs:
- If the networkElementId is null or it is not a valid network element InstanceNameId.
- If identityPolicyCol collection is null or empty.
- If identityPolicyCol collection has the existing identity policy name

Parameters
- opContext—Operational context
- networkElementId—InstanceNameId of network element in which the identity policies have to be created.
- identityPolicyCol—a collection of IdentityPolicy to be created.

Return Value
- void

createIdentityProfile

Creates an identity profile in a network element.

ValidationException is thrown if any of the following situation occurs:
- If the argument passed is null or it is not a valid network element InstanceNameId.
- If identityProfile is null or empty.

Parameters
- opContext—Operational context
- networkElementId—InstanceNameId of the network element
- identityProfile—IdentityProfile object

Return Value
- InstanceNameId of the created identity profile
createIpAdmissionControlRulesInNetworkElement

Creates the given IP Admission Control Rules in a Network Element.

ValidationException is thrown if any of the following situation occurs:
- If the argument passed is null or it is not a valid network element InstanceNameId.
- If ipAdmissionControlRules collection is null or empty.
- If ipAdmissionControlRules collection has the existing ip admission control name
- If ip admission control name length is more than 128 characters.

Parameters
opContext—Operational context
networkElementId—InstanceNameId of the network element in which rules will be created.

ipAdmissionControlRules—a collection of IpAdmission Control Rules

Return Value
A collection of InstanceNameIds of the created rules.

deleteAllAdmissionControlRulesInNetworkElement

Deletes all the IP admission control rules configured in a given network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
networkElementId—InstanceNameId of the network element

Return Value
void

deleteAllIdentityPoliciesInNetworkElement

Deletes all the Identity policies in a given network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
networkElementId—InstanceNameId of the network element in which identity policies have to be deleted.

Return Value
void
deleteIdentityPolicies

Deletes a given collection of Identity policies. ValidationException is thrown if any of the following situation occurs:

- If identityPolicyIdCol collection is null or it is empty.
- If identityPolicyIdCol collection contains an element that is not of type IdentityPolicy InstanceNameId.
- If identityPolicyIdCol collection contains a IdentityPolicy that does not exist in the database.

Parameters

- opContext—Operational context
- identityPolicyIdCol—InstanceNameIds of identity policies to be deleted.

Return Value

void

deleteIdentityProfileFromNetworkElement

Deletes an identity profile configured in a given network element. ValidationException is thrown if any of the following situation occurs:

- If identityProfileId is null or it is empty.
- If identityProfileId contains an element that is not of type IpAdmissionControlRule InstanceNameId.
- If identityProfileId contains a IpAdmissionControlRule that does not exist in the database.

Parameters

- opContext—Operational context
- identityProfileId—InstanceNameId of the identity profile to be deleted

Return Value

void

deleteIpAdmissionControlRules

Deletes the given collection of IP Admission Control Rules. ValidationException is thrown if any of the following situation occurs:

- If ipAdmissionControlRuleIdCol collection is null or it is empty.
- If ipAdmissionControlRuleIdCol collection contains an element that is not of type IpAdmissionControlRule InstanceNameId.
- If ipAdmissionControlRuleIdCol collection contains a IpAdmissionControlRule that does not exist in the database.
disableClientlessAuthenticationInNetworkElements

Disable clientless authentication feature in a collection of network elements.

ValidationException is thrown if any of the following situation occurs:
- If networkElementIdCol is null or it is empty.
- If networkElementIdCol contains invalid Network Element InstanceNameId or null value.
- If there is no equivalent Abstract Network Element object with the given InstanceNameId in the networkElementIdCol.

Parameters
- opContext—Operational context
- networkElementIds—a collection of InstanceNameId of network elements in which ClientlessAuthentication has to be disabled.

Return Value
void

disableIpDeviceTrackingInNetworkElements

Disables IP device tracking features in a collection of network elements.

ValidationException is thrown if any of the following situation occurs:
- If networkElementIdCol is null or it is empty.
- If networkElementIdCol contains invalid Network Element InstanceNameId or null value.
- If there is no equivalent Abstract Network Element object with the given InstanceNameId in the networkElementIdCol.

Parameters
- opContext—Operational context
- networkElementIdCol—a collection of InstanceNameIds of network elements in which device tracking feature has to be disabled.

Return Value
void
**disableNacService**

Disables Nac Service in a InstanceNameId network element. Service Enabling/Disabling is supported in DC OS platform. If this API is called with the network elements of Cat6k platform then FeatureException will be thrown.

ValidationException is thrown if any of the following situation occurs:

1. If neInstanceNameIdCol is null.
2. If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
3. If the network element does not exist in the database.

**Parameters**
- opContext—TODO
- neInstanceNameIdCol—A collection of InstanceNameId of the network elements.

**Return Value**
void

**enableClientlessAuthenticationInNetworkElements**

Enables clientless authentication feature in a collection of network elements.

ValidationException is thrown if any of the following situation occurs:

- If networkElementIdCol is null or it is empty.
- If networkElementIdCol contains invalid Network Element InstanceNameId or null value.
- If there is no equivalent Abstract Network Element object with the given InstanceNameId in the networkElementIdCol.

**Parameters**
- opContext—Operational context
- networkElementIdCol—a collection of InstanceNameId of network elements in which ClientlessAuthentication has to be enabled.

**Return Value**
void

**enableIpDeviceTrackingInNetworkElements**

Enables IP device tracking feature in a collection of network elements.

ValidationException is thrown if any of the following situation occurs:

- If networkElementIdCol is null or it is empty.
- If networkElementIdCol contains invalid Network Element InstanceNameId or null value.
- If there is no equivalent Abstract Network Element object with the given InstanceNameId in the networkElementIdCol.
enableNacService

Enables Nac Service in a InstanceNameId network element. Service Enabling/Disabling is supported in DC OS platform. If this API is called with the network elements of Cat6k platform then FeatureException will be thrown.

ValidationException is thrown if any of the following situation occurs:
1. If neInstanceNameIdCol is null.
2. If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
3. If the network element does not exist in the database.

Parameters
opContext—TODO
neInstanceNameIdCol— A collection of InstanceNameId of the network elements.

Return Value
void

getAdmissionControlRulesInNetworkElement

Returns all the IP admission control rules configured in a given network element.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
networkElementId—instance id of the given network element.

Return Value
A collection of IP Admission Control Rules. This collection will hold objects on type IpAdmissionControlRule

getAllNacHostSessionInNetworkElement

Returns the list of NAC host sessions in the network element.
Chapter 23      NacLpApp Service

getClientlessAuthenticationInNetworkElements

Returns the clientless authentication configurations done in a collection of network elements.
ValidationException is thrown if any of the following situation occurs:

- If `networkElementIdCol` is null or it is empty.
- If `networkElementIdCol` contains invalid Network Element InstanceNameId or null value.
- If there is no equivalent Network Element object with the given InstanceNameId in the `networkElementIdCol`.

Parameters

- `opContext`—Operational context
- `networkElementIdCol`—a collection of InstanceNameId of network elements.

Return Value

A collection of ClientlessAuthentication objects representing the clientless authentication feature configured in network element.

getEapOudpValidationSettingInInterfaces

Returns the EAPoUDP protocol parameters configured in a given collection of interfaces.
ValidationException is thrown if any of the following situation occurs:

- If `interfaceNameIds` is null or it is empty.
- If `interfaceNameIds` contains invalid Switched Network Interface InstanceNameId or null value.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId in the `interfaceNameIds`.

Parameters

- `opContext`—Operational context
- `interfaceNameIds`—a collection of InstanceNameId of switched network interfaces whose EAPoUDP parameters will be returned.

Return Value

A collection of EapOudpValidation objects representing the EAPoUDP parameters configured.
getExceptionListHostsInIdentityProfile

Gets a collection of exempted hosts associated with a given identity profile.
ValidationException is thrown if any of the following situation occurs:
- If identityProfileId is null or it is empty.
- If identityProfileId contains invalid identity profile InstanceNameId or null value.
- If there is no equivalent identity profile object with the given InstanceNameId in the identityProfileId.

Parameters
opContext—Operational context
identityProfile—InstanceNameId of identity profile

Return Value
A collection of exempted hosts associated with the identity profile. Returned collection will hold objects of type com.cisco.dcbu.dcm.model.nac.ExemptedHost

getIdentityPoliciesInNetworkElement

Returns a collection of identity policies configured in a given network element.
ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
networkElementId—instance id of network element

Return Value
A collection of identity policies configured in the network element. The returned collection will hold objects of type IdentityPolicy

getIdentityPolicyForExceptionListHost

Returns the identity policy assigned to a exempted list host.
ValidationException is thrown if any of the following situation occurs:
- If exemptHostInstanceNameIdCol is null or it is empty.
- If exemptHostInstanceNameIdCol contains invalid exempted list host InstanceNameId or null value.
- If there is no equivalent exempted list host object with the given InstanceNameId in the exemptHostInstanceNameIdCol.

Parameters
opContext—& param exemptHostInstanceNameIdCol instance id of ExemptedHost.
getIdentityProfilesInNetworkElement

Returns all the identity profiles configured in a network element. Returned list size will be one for DC3, since DCOS supports only EapoUdp.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
networkElementId—InstanceNameId of network element

Return Value
A collection of identity profiles configured in the given network element. Collection returned will hold objects of type IdentityProfile

getInterfacesUsingIpAdmissionControlRule

Returns a collection of interfaces on which a given IP admission control rule is applied.

ValidationException is thrown if the argument passed is null or it is not a valid ip admission control rule InstanceNameId.

Parameters
opContext—Operational context
ipAdmissionControlRule—InstanceNameId of IP admission control rule

Return Value
A collection of interfaces on which the given IP admission control is applied. This collection will have objects of type SwitchedNetworkInterface

getIpAdmissionControlRuleAppliedOnInterfaces

Returns the collection of IP Admission Control Rule applied on interfaces. This API is applicable for Catalyst 6500 switch and is not applicable for DC3. For DC3, this API throws validation exception. If a particular interface does not have IP Admission control rule, the API populates NULL value in the returned collection for that interface.

ValidationException is thrown if the argument passed is null or it is not a valid switched network interface InstanceNameId.

Parameters
opContext—Operational context
interfaceInstanceIds—collection of InstanceNameId of interfaces
getIpDeviceTrackingInNetworkElements

Returns the IP device tracking configurations done in a collection of network elements.
ValidationException is thrown if any of the following situation occurs:
- If networkElementIdCol is null or it is empty.
- If networkElementIdCol contains invalid Network Element InstanceNameId or null value.
- If there is no equivalent Abstract Network Element object with the given InstanceNameId in the networkElementIdCol.

Parameters
opContext—Operational context
networkElementIdCol—a collection of InstanceNameId of network elements

Return Value
A collection of IpDeviceTracking objects in network elements.

getLpIpGlobalSettingsInNetworkElements

Returns the LPIP Global Settings configured in a collection of network elements.
ValidationException is thrown if any of the following situation occurs:
- If networkElementIdCol is null or it is empty.
- If networkElementIdCol contains invalid Network Element InstanceNameId or null value.
- If there is no equivalent Abstract Network Element object with the given InstanceNameId in the networkElementIdCol.

Parameters
opContext—Operational context
networkElementIdCol—a collection of InstanceNameId of network elements

Return Value
A collection of LpIpGlobalSetting objects representing the global LPIP settings configured in a network element.

getLpIpTrackedDevicesInNetworkElement

Returns a list of LPIP tracked devices.
getLpIpTrackedDevicesInSwitchedNetworkInterface

Returns a list of LPIP tracked devices in the switched network interface.

Parameters
opContext—Operational context
networkElementId—a network element ID of a switched network interface

Return Value
A list of LpIpTrackedDeviceStatus.

getNacHostSessionInSwitchedNetworkInterface

Returns a list of LPIP tracked devices.

Parameters
opContext—Operational context
networkElementId—a network element ID for a switched network interface.

Return Value
A list of NacHostSession for the switched network interface.

modifyClientlessAuthentication

Updates the server with modified clientless authentication configurations.
ValidationException is thrown if any of the following situation occurs:
• If the networkElementId is null or it is not a valid network element InstanceNameId.
• clientlessAuthentication object is null or empty

Parameters
opContext—Operational context
networkElementId—InstanceNameId of the network element
clientlessAuthentication—Modified ClientlessAuthentication object ClientlessAuthentication
modifyExceptionListHostsInIdentityProfile

Modifies a given collection collection of ExceptionListHosts configured in an identity profile. ValidationException is thrown if any of the following situation occurs:

- If identityProfileId is null or it is empty.
- If identityProfileId contains an element that is not of type IdentityProfile InstanceNameId.
- If identityProfileId contains a IdentityProfile that does not exist in the database.
- If exceptionListHostCol is null or empty.

Parameters

opContext—Operational context
identityProfileId—InstanceNameId of Identity Profile.
exceptionListHostCol—a collection of ExceptionListHost objects that are modified

Return Value

void

modifyIdentityPolicies

Modifies a given collection of identity policies. ValidationException is thrown if any of the following situation occurs:

- If identityPolicyCol collection is null or it is empty.
- If identityPolicyCol collection contains a IdentityPolicy that does not exist in the database.

Parameters

opContext—Operational context
identityPolicyCol—a collection of IdentityPolicy that are modified by the client.

Return Value

void

modifyIdentityProfiles

Modifies a collection of identity profiles. This modification will address addition, removal and modification of ExceptionListHosts bound to an identity profile. This modification will also address Identity Policy association to each ExceptionListHost. There will be only one IdentityProfile of type EAPoUDP in a network element. Each identity profile in the argument will be corresponding to a different network element.
ValidationException is thrown if any of the following situation occurs:

- If `identityProfileCol` collection is null or it is empty.
- If `identityProfileCol` collection contains a `IdentityProfile` that does not exist in the database.

### Parameters

- **opContext**—Operational context
- **identityProfileCol**—a collection of `IdentityProfile` in a network element. Each identity profile corresponds to a different network element.

### Return Value

`void`

## modifyIpAdmissionControlRules

Modifies the given collection of IP Admission Control Rules. ValidationException is thrown if any of the following situation occurs:

- if the `ipAdmissionControlRules` Collection is null or empty
- If `identityPolicyCol` collection contains a `IdentityPolicy` that does not exist in the database.

### Parameters

- **opContext**—Operational context
- **ipAdmissionControlRuleCol**—a collection of `IpAdmissionControlRule` to be modified.

### Return Value

`void`

## modifyIpDeviceTracking

Updates the server with modified device tracking configurations. ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid network element `InstanceNameId`
- `ipDeviceTracking` object is null or empty

### Parameters

- **opContext**—Operational context
- **networkElementId**—`InstanceNameId` of the network element.
- **ipDeviceTracking**—Modified `IpDeviceTracking` object `IpDeviceTracking`

### Return Value

`void`
modifyLpIpGlobalSettingsInNetworkElements

Modifies a given collection of LPIP global settings.

ValidationException is thrown if any of the following situation occurs:

- If lpIpGlobalSettingCol collection is null or it is empty.
- If lpIpGlobalSettingCol collection contains an object that is not of type LpIpGlobalSetting.

PropertiesException is thrown if any of the following situation occurs:

- In the lpIpGlobalSettingCol collection, if any attribute in the LpIpGlobalSetting is not valid.

Parameters

- opContext—Operational context
- lpIpGlobalSettingCol—a collection of LpIpGlobalSetting objects modified

Return Value

void

removeExceptionListHostsFromIdentityProfile

Removes a Exception List host from an identity profile.

ValidationException is thrown if any of the following situation occurs:

- If exemptListHostIdCol collection is null or it is empty.
- If exemptListHostIdCol collection contains an element that is not of type ExceptionListHost InstanceNameId.
- If exemptListHostIdCol collection contains a ExceptionListHost that does not exist in the database.
- If identityProfileId is null or it is empty.
- If identityProfileId contains an element that is not of type IdentityProfile InstanceNameId.
- If identityProfileId contains a IdentityProfile that does not exist in the database.

Parameters

- opContext—Operational context
- identityProfileId—InstanceNameId of IdentityProfile object
- exceptionListHosts—a collection of InstanceNameId of ExceptionListHost objects

Return Value

void

unbindIdentityPolicyFromExceptionListHosts

Clears an identity policy assigned to a Exception List host.

ValidationException is thrown if any of the following situation occurs:

- If exceptionLishHostIdCol collection is null or it is empty.
If exceptionListHostIdCol collection contains an element that is not of type ExceptionListHost InstanceNameId.
If exceptionListHostIdCol collection contains a ExceptionListHost that does not exist in the database.

Parameters

- opContext—Operational context
- exceptionListHostIdCol—InstanceNameId of the ExceptionListHost object

Return Value

- void
ObjectTrackingApp Service

This chapter describes the DCNM web services’ API methods for the ObjectTrackingApp service.

Information About ObjectTrackingApp Service

Object tracking allows you to track the state of an interface or an IP route that will be used by client processes such as HSRP or the Gateway Local Balancing Protocol (GLBP). A client process, such as GLBP, can register an interest in tracking objects and request notification when the tracked object changes its state. This feature increases the availability and recovery speed of a routing system and decreases outages and the outage duration. The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Modify APIs—Modify existing Network Interface Tracking and Route Tracking settings.
- Delete APIs—Delete a tracking instance in the device.

createTrackedObjectsInNetworkElements

Creates a list object Tracking setting in the given network element. ValidationException is thrown if any of the following situation occurs:

- If objectTrackingCol is null or empty or it is not of type ObjectTrackingSetting.

Parameters:
- opContext—Operational context
- neInstanceNameId—InstanceNameId of AbstractNetworkElement
- objectTrackingCol—List of ObjectTrackingSetting objects

Return Value:
List of InstanceNameIds of ObjectTrackingSetting

deleteTrackedObjects

Deletes one or more Object Tracking Setting given instanceNameIds of ObjectTrackingSetting
**getTrackedClients**

Returns a list of GlbpGroupSetting Objects processing the given trackingId. 

**Parameters**
- opContext—Operational context
- trackingId—InstanceNameId of ObjectTrackingSetting

**Return Value**
- A collection of GlbpGroupSetting objects. The returned objects will have their corresponding IpNetworkInterface reference. But If that IpNetworkInterface has any other associations, those will be cleared except NetworkInterfaceName.

**getTrackedObjects**

Returns the one or more ObjectTrackingSetting setting. Given the InstanceNameId of one or more ObjectTrackingSetting, returns the corresponding object tracking setting.

**Parameters**
- opContext—Operational context
- trackingInstanceNameIdCol—InstanceNameId of one or more ObjectTrackingSetting

**Return Value**
- List of ObjectTrackingSetting. In the returned list of objects, only the following associations will be present and all other associations will be cleared.
  - AbstractNetworkElement for ObjectTrackingSetting
getTrackedObjectsInNetworkElement

Returns a collection of ObjectTrackingSetting configured in the specified network elements. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or it is not of type InstanceNameId.
- If neInstanceNameIdCol is not a valid object tracking InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—InstanceNameId of one or more AbstractNetworkElement

Return Value

A collection of ObjectTrackingSetting. In the returned list of objects, only the following associations will be present and all other associations will be cleared.

- AbstractNetworkElement for ObjectTrackingSetting
- NetworkInterface association for NetworkInterfaceTrackingSetting
- Vrf association for RouteTrackingSetting
- ObjectTrackingStatus association for ObjectTrackingSetting

modifyTrackedObjects

Modifies one or more existing Tracked Objects. ValidationException is thrown if any of the following situation occurs:

- If trackedObjCol is null or empty or it is not of type ObjectTrackingSetting.
- Only existing object tracking setting (exist in database) can be modified.

Parameters

- opContext—Operational context
- trackedObjCol—List of modified ObjectTrackingSetting objects

Return Value

List of modified ObjectTrackingSetting persisted in the database. In the returned list of objects, only the following associations will be present and all other associations will be cleared.

- AbstractNetworkElement for ObjectTrackingSetting
- NetworkInterface association for NetworkInterfaceTrackingSetting
- Vrf association for RouteTrackingSetting
- ObjectTrackingStatus association for ObjectTrackingSetting
PollerApp Service

This chapter describes the DCNM web services’ API methods for the PollerApp service.

Information About PollerApp Service

The PollerApp service allows you to obtain syslog poller information. The syslog poller component is used to synchronize with the device automatically. The DCNM server creates a syslog poller for each managed device. Each syslog poller periodically retrieves system and accounting logs from a device. DCNM uses the information retrieved by poller processes to update its configuration and status information for polled devices.

getAllPollerTasks

Returns all the SyslogPollerInfo running in the server.

Parameters

opContext—Operational context

Return Value

List of SyslogPollerInfo for all syslog poller tasks.

getLoggingLevelSettings

Returns a collection of LoggingLevelSetting objects for given devices.

Parameters

opcontext—Operational context
neInstanceNameIdCol—InstanceNameId of AbstractNetworkElement

Return Value

Collection of LoggingLevelSetting objects corresponding to given devices.
getLoggingLevelSettingsForDcnmDefaults

Returns a collection of LoggingLevelSetting that are set to DCNM required values based on the given platform type and version.

Parameters

- opContext—Operational context
- platformTypeVersionPairCol—Collection of a pair of platform type and the version in it.

Return Value

- Returns a collection of LoggingLevelSetting objects

getPollerInterval

Returns the interval of the syslog poller in seconds.

Parameters

- opContext—

Return Value

- The polling interval in seconds

modifyLoggingLevelSettings

Modifies the loggingLevelSetting object's value to be the given LoggingLevelSettings.

Parameters

- opContext—Operational context
- loggingLevelSettingCol—Collection of modified LoggingLevelSetting objects.

Return Value

- Modified LoggingLevelSetting objects.

modifyLoggingLevelSettingsToDcnmDefaults

Sets the log levels in given devices to be DCNM required default values.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—Collection of Instance name id of AbstractNetworkElement.

Return Value

- void
modifyPollerInterval

Modifies the given number of seconds as the polling interval of the syslog poller.

**Parameters**

- opContext—Operational context
- seconds—polling interval in seconds

**Return Value**

void

startSyslogPoller

Starts the syslog polling for the given network element id.

**Parameters**

- opContext—Operational context
- networkElementId—network element id for which the polling has to be started.

**Return Value**

void

stopSyslogPoller

Stops the syslog polling for the given network element.

**Parameters**

- opContext—Operational context
- networkElementId—network element for which the syslog polling has to be stopped.

**Return Value**

void

synchronizeWithDevice

Syncronizes with the device. Runs syslog polling of the given network element once immediately to syncronize the changes done in the device with DCNM.

**Parameters**

- opContext—Operational context
- networkElementId—network element for which the syslog polling has to be run immediately once.
**validateLoggingLevelSettingsForDcnmDefaults**

Validate the logging level setting to check if it's attributes are set to DCNM required defaults for the given devices.

**Parameters**
- **opContext**—Operational context
- **neInstanceNameIdCol**—Collection of Instance name id of AbstractNetworkElement.

**Return Value**
Returns a collection Boolean objects. It is set to true if the device is configured with DCNM required logging level, else set to false.
PortSecurityApp Service

This chapter describes the DCNM web services’ API methods for the PortSecurityApp service.

Information About PortSecurityApp Service

The port security feature allows you to restrict input to an interface by limiting and identifying MAC addresses of the workstations that are allowed to access the port. Port security can be enabled only on switched network interfaces. The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Modify APIs—Modify existing port security parameters.
- Bind and Unbind APIs—Bind and unbind secured MAC address.
- Enable and Disable APIs—Enable and disable the port security.

bindPortSecuritySettingToNetworkInterfaces

Applies a given PortSecurityNetworkInterfaceSetting to a collection of interfaces. This collection of interfaces shall have both port security enabled and disabled interfaces.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameIds is null or it is empty.
- If interfaceNameIds contains invalid Switched Network Interface InstanceNameId or null value.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId in the interfaceNameIds.
- If portSecurityNetworkInterfaceSetting is null

Parameters

- opContext—Operational context
- interfaceNameIds—a collection of InstanceNameId of switched network interfaces to which a port security setting has to be applied.

Return Value

void
bindStaticSecureHostsToAccessNetworkInterface

Assigns a collection of hosts statically to a switched network interface operating in access mode. ValidationException is thrown if any of the following situation occurs:

- If interfaceNameId is null.
- If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId
- If securedStaticMacAddressCol collection is null or it is empty.

Parameters

opContext—TODO
interfaceNameId—InstanceNameId of the switched network interface.
secureStaticMacAddressCol—a collection of SecuredMacAddress to be bound to the interface.

Return Value

A collection of InstanceNameId of secured mac address.

bindStaticSecureHostsToPvlanHostNetworkInterface

Assigns a collection of hosts statically to a switched network interface operating in Private VLAN Host mode. ValidationException is thrown if any of the following situation occurs:

- If interfaceNameId is null.
- If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId
- If securedStaticMacAddressCol collection is null or it is empty.

Parameters

opContext—Operational context
interfaceNameId—InstanceNameId of the switched network interface.
secureStaticMacAddressCol—a collection of SecuredMacAddress to be bound to the interface.

Return Value

A collection of InstanceNameId of secured mac address.

bindStaticSecureHostsToPvlanPromiscousNetworkInterface

Assigns a collection of hosts statically to a switched network interface operating in Private VLAN promiscuous mode. ValidationException is thrown if any of the following situation occurs:

- If interfaceNameId is null.
bindStaticSecureHostsToVlanInTrunkNetworkInterface

Assigns a collection of hosts statically to a switched network interface operating in trunk mode. This will support the VLANs which does not exist on the device (preprovisioning).

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameId is null.
- If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId
- If securedStaticMacAddressCol collection is null or it is empty.

Parameters

- opContext—Operational context
- interfaceNameId—InstanceNameId of the switched network interface.
- vlanIds—a collection of InstanceNameId of the VLANs which is carried by trunk network interface.
- secureStaticMacAddressCol—a collection of SecuredMacAddress bound to the interface.

Return Value

A collection of InstanceNameId of secured mac address.
clearPortSecuritySettingToNetworkInterfaces

Clears the port security configurations in collection of interfaces.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameIds is null or it is empty.
- If interfaceNameIds contains invalid Switched Network Interface InstanceNameId or null value.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId in the interfaceNameIds.

Parameters

opContext—Operational context
interfaceNameIds—a collection of InstanceNameId of switched network interfaces to which a port security setting has to be applied.

Return Value

void

clearPortSecuritySettingToNetworkInterfaces

Returns a list of newly created PortSecurityNetworkInterfaceSettingForVlan. Only maxAddress is configured for this newly created class.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameId is null or empty.
- If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
- If maxAddressList is null.
- If the size of the vlanRange is not equal to the size of the maxAddressList.
- In case of the accessPort, if the VLAN size is more than one.

Parameters

interfaceNameId—InstanceNameId of the switched network interface operating in trunk mode.
vlanRange—Range of VLANs (includes comma separated and hyphenated VLAN IDs)
secureStaticMacAddressCol—a collection of SecuredMacAddress bound to the interface.

Return Value

A collection of InstanceNameId of secured mac address.
### deletePortSecurityNetworkInterfaceSettingForVlans

Deletes one or more standard Port Security Network Interface Setting for VLAN objects. Given the InstanceNameId of the PortSecurityNetworkInterfaceSettingForVlan objects, those objects will be deleted from the server. ValidationException is thrown if any of the following situation occurs:

- If psSettingForVlanInstanceNameIds is null or it is empty.
- If psSettingForVlanInstanceNameIds contains invalid Port Security Network Interface Setting for VLAN InstanceNameId or null value.
- If there is no equivalent PortSecurityNetworkInterfaceSettingForVlan object with the given InstanceNameId in the psSettingForVlanInstanceNameIds.

**Parameters**

- opContext—Operational context
- psSettingForVlanInstanceNameIds—InstanceNameId of the PortSecurityNetworkInterfaceSettingForVlan.

**Return Value**

void

### disablePortSecurityInAllInterfacesInNetworkElement

Disables port security in all the interfaces in a network element. ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

**Parameters**

- opContext—Operational context
- networkElementId—InstanceNameId of the network element.

**Return Value**

void
disablePortSecurityInNetworkInterfaces

Disables port security in a given set of interfaces which have port security enabled.

ValidationException is thrown if any of the following situation occurs:

- If interfaceName is null.
- If interfaceName is not a valid Switched Network Interface InstanceNameId.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId

Parameters

- opContext—Operational context
- interfaceNames—a collection of InstanceNameId of switched network interfaces in which port security is enabled.

Return Value

void

disablePortSecurityService

Disables PortSecurity Service in a InstanceNameId network element. Service Enabling/Disabling is supported in DC OS platform. If this API is called with the network elements of Catalyst 6500 series switches then FeatureException will be thrown.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null.
- If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—— A collection of InstanceNameId of the network elements.

Return Value

void

disablePortSecurityStopLearning

Disables PortSecurity Stop Learning in a InstanceNameId network element. Stop Learning Enabling/Disabling is supported in DC OS platform. If this API is called with the network elements of Catalyst 6500 series switches then FeatureException will be thrown.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null.
- If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
- If the network element does not exist in the database.
enablePortSecurityInNetworkInterfaces

Enables port security in a given set of interfaces. Port Security can be enabled in the following interfaces.
- Port security disabled Switched Network interfaces.
- Switched Network Interfaces which don't have port security configurations.

ValidationException is thrown if any of the following situation occurs:
- If interfaceName is null.
- If interfaceName is not a valid Switched Network Interface InstanceNameId.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId.

Parameters
- `opContext`—Operational context
- `neInstanceNameIdCol`—A collection of InstanceNameId of the network elements.

Return Value
- void

enablePortSecurityService

Enables PortSecurity Service in a InstanceNameId network element. Service Enabling/Disabling is supported in DC OS platform. If this API is called with the network elements of Catalyst 6500 series switches then FeatureException will be thrown.

ValidationException is thrown if any of the following situation occurs:
- 29. If `neInstanceNameIdCol` is null.
- 30. If `neInstanceNameIdCol` does not contain a valid network element InstanceNameId.
- 31. If the network element does not exist in the database.

Parameters
- `opContext`—Operational context
- `neInstanceNameIdCol`—A collection of InstanceNameId of the network elements.

Return Value
- void
enablePortSecurityStopLearning

Enables PortSecurity Stop Learning in a InstanceNameId network element. Stop Learning Enabling/Disabling is supported in DC OS platform. If this API is called with the network elements of Catalyst 6500 series switches then FeatureException will be thrown.

ValidationError is thrown if any of the following situation occurs:
35. If neInstanceNameIdCol is null.
36. If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
37. If the network element does not exist in the database.

Parameters
opContext—Operational context
neInstanceNameIdCol—A collection of InstanceNameId of the network elements.

Return Value
void

getAllDynamicSecureMacAddressesInNetworkInterface

Returns all the host MAC addresses that are learnt dynamically in all the VLANs in a switched network interface. For trunk ports, MAC addresses bound to the port through all the VLANs will be returned. For access ports, MAC addresses bound to the access VLAN will be returned. ValidationException is thrown if the argument passed is null or it is not a valid Switched Network Interface InstanceNameId.

Parameters
opContext—Operational context
interfaceNameId—InstanceNameId of the switched network interface.

Return Value
void

getAllDynamicSecureMacAddressesInVlanOfNetworkInterface

Returns all the host MAC addresses that are learnt dynamically in a VLAN in a switched network interface.

ValidationException is thrown if any of the following situation occurs:
• If interfaceNameId is null.
• If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
• If vlanId is null.
• If vlanId is not a valid VLAN InstanceNameId.
getAllDynamicSecureMacAddressesInVlanOfNetwork InterfaceByVlanId

Returns all the host MAC addresses that are learnt dynamically in VLANs in a switched network interface. This method will address assigning static secure MAC address in a VLAN that has not been created (preprovisioning).

ValidationException is thrown if any of the following situation occurs:
- If interfaceNameId is null.
- If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
- If vlanId is null.

Parameters
- opContext—Operational context
- interfaceNameId—InstanceNameId of the switched network interface.
- VLAN—Valid VLAN Id.

Return Value
A collection of SecuredDynamicMacAddress objects pertaining to the hosts that are bound the given VLAN.

getAllPortSecurityDisabledInterfacesInNetworkElement

Returns all the port security disabled interfaces in a given network element. Given the instance name ID of the network element, returns a collection switched network interface objects. The following are the criteria to check the port security disabled interfaces.
- PortSecurityNetworkInterfaceSetting parameters are configured and getPortSecurityEnable() method returns false.
- PortSecurityNetworkInterfaceSetting parameters are not configured and the collection size of PortSecuritySetting size is greater than 0.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
getAllPortSecurityEnabledInterfacesInNetworkElement

Returns all the port security enabled interfaces in a given network element. Given the instance name ID of the network element, returns a collection switched network interface objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId of the network element.

Return Value
A collection of port security enabled SwitchedNetworkInterfaces. only the following associations will be present, and all other associations will be cleared.

- PortSecurityNetworkInterfaceSetting
- NetworkInterfaceName
- PortSetting
- PortCapability
- NetworkInterfaceStatus
- PortStatus
- AbstractNetworkElement
- Collection of RoutedSubNetworkInterface
- Collection of Switched Network Interfaces belongs to VLAN Collection
Chapter 26 PortSecurityApp Service

getAllPortSecurityVlanSettingsForNetworkInterface

Returns the port security settings configured in each VLAN in a collection of switched network interfaces. For trunk ports, port security settings of all the VLANs in the trunk will be returned. For access ports, port security settings of the access VLAN will be returned.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameIds is null or it is empty.
- If interfaceNameIds contains invalid Switched Network Interface InstanceNameId or null value.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId in the interfaceNameIds.

Parameters

- opContext—Operational context
- interfaceNameIds—a collection of InstanceNameIds InstanceNameId of the switched network interfaces.

Return Value

A collection of PortSecurityNetworkInterfaceSettingForVlan objects pertaining to the VLANs in the interface. For access ports, size of the returned collection will be 1.

getAllStaticSecureMacAddressesInNetworkInterface

Returns all the host MAC addresses that are bound statically to all the VLANs in a switched network interface. For trunk ports, MAC addresses bound to the port through all the VLANs will be returned. For access ports, MAC addresses bound to the access VLAN will be returned.

ValidationException is thrown if the argument passed is null or it is not a valid Switched Network Interface InstanceNameId.

Parameters

- opContext—Operational context
- interfaceNameId—InstanceNameId of the switched network interface.

Return Value

A collection of SecureStaticMacAddress objects pertaining to the hosts that are bound to the Switched Network Interface.

getAllStaticSecureMacAddressesInVlanOfNetworkInterface

Returns all the host MAC addresses that are bound statically to a VLAN in a switched network interface. This method will get the static secure MAC address in a VLAN that had been created.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameId is null.
- If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
getAllStaticSecureMacAddressesInVlanOfNetworkInterface ByVlanId

If vlanId is null.
If vlanId is not a valid VLAN InstanceNameId.

Parameters
opContext—Operational context
interfaceNameId—InstanceNameId of the switched network interface.
vlanId—InstanceNameId of the VLAN.

Return Value
A collection of SecureStaticMacAddress objects pertaining to the hosts that are bound to the given VLAN.

getAllStaticSecureMacAddressesInVlanOfNetworkInterface ByVlanId

Returns all the host MAC addresses that are bound statically to a VLANs in a switched network interface. This method will address assigning static secure MAC address in a VLAN that has not been created.

ValidationException is thrown if any of the following situation occurs:
• If interfaceNameId is null.
• If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
• If vlanId is null.

Parameters
opContext—Operational context
interfaceNameId—InstanceNameId of the switched network interface.
VLAN—Valid VLAN Id.

Return Value
A collection of SecureStaticMacAddress objects pertaining to the hosts that are bound the given VLAN.

getPortSecurityCapableInterfaces

Returns all the port security capable interfaces in a given network element and a slot number. Given the instance name ID of the network element and slot no, returns a collection port security capable switched network interface objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
networkElementInstanceNameId—InstanceNameId of the network element.
getPortSecurityGlobalSettingsInNetworkElements

Returns the port security global configurations configured in a given list of network elements. List of network elements can include Virtual Network Elements also. This API is applicable only for Nexus 7000 series switch Platform. ValidationException is thrown if any of the following situation occurs:

- If abstractNetworkElementIds is null or it is empty.
- If abstractNetworkElementIds contains invalid Abstract Network Element InstanceNameId or null value.
- If there is no equivalent Abstract Network Element object with the given InstanceNameId in the abstractNetworkElementIds.

Parameters

- opContext—Operational context
- abstractNetworkElementIds—a collection of InstanceNameIds of Abstract Network Elements.

Return Value

A collection of PortSecurityGlobalSetting objects representing the global Port Security configurations done in the Abstract Network Elements.

cgetPortSecurityServiceStateInNetworkElements

Returns state of PortSecurity Service like whether PortSecurityService is enabled or disabled in a list of network elements. Given the list of instance name IDs of the network elements, returns a array of Boolean values. ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null
- If neInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId
- if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—InstanceNameId of the one or more Network Element for which the DhcpSnoopingService state is required

Return Value

The returned array will contain Boolean instances.

Boolean value TRUE indicates PortSecurityService is enabled in the given network element. Boolean value FALSE indicates PortSecurityService is disabled in the given network element.
getPortSecuritySettingsInNetworkInterfaces

Returns the port security configurations done in a given collection of network interfaces. This collection shall have port security settings for both port security enabled and port security disabled interfaces. This collection shall have null in case of interface doesn’t have the port security setting parameters.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameIds is null or it is empty.
- If interfaceNameIds contains invalid Switched Network Interface InstanceNameId or null value.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId in the interfaceNameIds.

Parameters

opContext—Operational context

interfaceNameIds—a collection of InstanceNameId of switched network interfaces.

Return Value

A collection of PortSecurityNetworkInterfaceSetting objects pertaining to all the SwitchedNetworkInterfaces.

getPortSecurityStopLearningStateInNetworkElements

Returns state of PortSecurity stop learning state like whether PortSecurityStopLearning is enabled or disabled in a list of network elements. Given the list of instance name IDs of the network elements, returns a array of Boolean values.

ValidationException is thrown if any of the following situation occurs:

- if the argument passed is null
- If neInstanceNameIdCol collection contains an element that is null or the collection is empty or it is not type InstanceNameId.
- if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

opContext—Operational context

neInstanceNameIdCol—InstanceNameId of the one or more Network Element for which the DhcpSnoopingService state is required

Return Value

The returned array will contain Boolean instances.

Boolean value TRUE indicates PortSecurityStopLearning is enabled in the given network element.

Boolean value FALSE indicates PortSecurityStopLearning is disabled in the given network element.
getPortSecurityVlanSettingsForNetworkInterface

Returns the port security settings configured in a given collections of VLANs in a switched network interface. This object will always have a collection of SecuredMacAddress objects and the SecuredDynamicMacAddress objects.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameId is null.
- If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
- If vlanIds are null or it is empty.
- If interfaceNameIds contains invalid VLAN InstanceNameId or null value.
- If there is no equivalent VLAN the given InstanceNameId in the vlanIds.

Parameters

- opContext—Operational context
- interfaceNameId—InstanceNameId of the switched network interface.
- vlanIds—a collection of InstanceNameId of the VLANs that are members of the trunk. For access ports, this collection contains only the InstanceNameId of the access VLAN.

Return Value

A collection of PortSecurityNetworkInterfaceSettingForVlan objects pertaining to the given VLANs in the interface. For access ports, size of the returned collection will be 1.

getPortSecurityVlanSettingsForNetworkInterfaceByRange

Returns the port security settings configured in a given collections of VLANs represented by vlanRange (Preprovisioning). This object will always have a collection of SecuredMacAddress objects and the SecuredDynamicMacAddress Object.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameId is null.
- If interfaceNameId is not a valid Switched Network Interface InstanceNameId.
- If vlanRange is null or it is empty.

Parameters

- opContext—Operational context
- interfaceNameId—InstanceNameId of the switched network interface.
- vlanRange—Range of VLANs (includes comma separated and hyphenated VLAN IDs)

Return Value

A collection of PortSecurityNetworkInterfaceSettingForVlan objects pertaining to the VLANs in the interface. For access ports, size of the returned collection will be 1.
modifyPortSecurityGlobalSettings

Updates the given list of port security global settings applied on network elements. This API is specific to Nexus 7000 series switch platform.

ValidationException is thrown if any of the following situation occurs:

- If portSecurityGlobalSetting collection is null or it is empty.
- If portSecurityGlobalSetting collection contains an object that is not of type PortSecurityGlobalSetting.

Parameters

- opContext—Operational context
- portSecurityGlobalSettings—a collection of PortSecurityGlobalSetting Objects. networkElementIds

Return Value

void

modifyPortSecuritySettingInNetworkInterfaces

Updates the server with a collection of modified PortSecurityNetworkInterfaceSetting Objects.

ValidationException is thrown if any of the following situation occurs:

- If portSecurityNetworkInterfaceSetting collection is null or it is empty.
- If portSecurityNetworkInterfaceSetting collection contains an object that is not of type PortSecurityNetworkInterfaceSetting.

PropertiesException is thrown if any of the following situation occurs:

- In the portSecurityNetworkInterfaceSetting collection, if any attribute in the PortSecurityNetworkInterfaceSetting is not valid.

IntegrityException is thrown if any of the following situation occurs:

- If the portSecurityNetworkInterfaceSetting collection contains a PortSecurityNetworkInterfaceSetting that does not exist in the database.

Parameters

- opContext—Operational context
- portSecurityNetworkInterfaceSettingCol—a collection of modified PortSecurityNetworkInterfaceSetting objects

Return Value

void

modifyPortSecurityVlanSettingsInNetworkInterface

Updates the server with a collection of modified PortSecurityNetworkInterfaceSettingForVlan objects.

ValidationException is thrown if any of the following situation occurs:
unbindAllSecureDynamicHostsFromNetworkInterface

Clears all the hosts dynamically learnt in a network interface. In case of access port, this method will unbind all the hosts learnt dynamically in access VLAN. In case of trunk ports, this method will unbind all the hosts learnt dynamically in all VLANs of the trunk.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameIds is null or it is empty.
- If interfaceNameIds contains invalid Switched Network Interface InstanceNameId or null value.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId in the interfaceNameIds.

Parameters
- opContext—Operational context
- interfaceNameIds—a collection of InstanceNameId of interfaces from which static secure hosts have to be cleared.

Return Value
void

unbindAllSecureDynamicHostsFromVlanTrunkInNetwork Interface

Clears all hosts dynamically learnt in a collection of VLANs in a trunk network interface.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameIds is null or it is empty.
- If interfaceNameIds contains invalid Switched Network Interface InstanceNameId or null value.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId in the interfaceNameIds.

Parameters
- opContext—Operational context
- interfaceNameIds—a collection of InstanceNameId of interfaces from which static secure hosts have to be cleared.

Return Value
void
unbindAllSecureDynamicHostsFromVlanTrunkInNetwork InterfaceByRange

Clears all hosts bound to a collection of VLANs allowed in a trunk network interface. This method will address clearing secure Dynamic MAC address in a VLAN that has not been created (preprovisioning). ValidationException is thrown if any of the following situation occurs:

- If interfaceNameId is null or it is empty.
- If interfaceNameId contains invalid Switched Network Interface InstanceNameId or null value.
- If vlanRange is null or it is empty.

Parameters
opContext—Operational context
interfaceNameId—InstanceNameId of the interface.
vlanRange—a comma separated or hyphenated string representing a collection of VLAN IDs.

Return Value
void

unbindAllStaticSecureHostsFromNetworkInterface

Clears all the static secure hosts bound to a network interface. In case of Access ports, this method will clear all the hosts bound to access VLAN. In case of trunk ports, this method will clear hosts bound to all VLANs of the trunk.

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameIds is null or it is empty.
- If interfaceNameIds contains invalid Switched Network Interface InstanceNameId or null value.

Parameters
opContext—Operational context
interfaceNameIds—a collection of InstanceNameIds of VLANs.

Return Value
void
unbindAllStaticSecureHostsFromVlanInTrunkInNetwork Interface

Clears all hosts bound to a collection of VLANs in a trunk network interface. ValidationException is thrown if any of the following situation occurs:

- If interfaceNameIds is null or it is empty.
- If interfaceNameIds contains invalid Switched Network Interface InstanceNameId or null value.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId in the interfaceNameIds.
- If vlanIds collection is null or it is empty.
- If vlanIds collection contains an object that is not of type VlanExternal.

Parameters

- opContext—Operational context
- interfaceNameIds—InstanceNameId of the interface.
- vlanIds—a collection of InstanceNameIds of VLANs.

Return Value

void

unbindAllStaticSecureHostsFromVlanInTrunkInNetwork InterfaceByRange

Clears all hosts bound to a collection of VLANs allowed in a trunk network interface. This is used to handle non-existing VLANs (preprovisioning).

ValidationException is thrown if any of the following situation occurs:

- If interfaceNameIds is null or it is empty.
- If interfaceNameIds contains invalid Switched Network Interface InstanceNameId or null value.
- If there is no equivalent Switched Network Interfaces object with the given InstanceNameId in the interfaceNameIds.

Parameters

- opContext—Operational context
- interfaceNameId—InstanceNameId of the interface.
- vlanIds—a collection of InstanceNameIds of VLANs.

Return Value

void
unbindSecureDynamicHostsFromVlanInTrunkNetwork Interface

Clears a collection of hosts dynamically learnt in a VLAN in a trunk network interface.
ValidationException is thrown if any of the following situation occurs:
- If secureDynamicMacAddress collection is null or it is empty.

Parameters
opContext—Operational context
secureDynamicMacAddressCol—a collection of SecuredDynamicMacAddress bound to the interface.

Return Value
void

unbindStaticSecureHostsFromVlanInTrunkNetworkInterface

Clears a collection of hosts bound to a trunk network interface.
ValidationException is thrown if any of the following situation occurs:
- If securedStaticMacAddress collection is null or it is empty.

Parameters
opContext—Operational context
secureHostInstanceNameIdCol—a collection of InstanceNameId of SecureStaticMacAddress bound to the interface

Return Value
void
CHAPTER 27

RbacApp Service

This chapter describes the DCNM web services’ API methods for the RbacApp service.

Information About RbacApp Service

Role-based access control (RBAC) allows you to restrict system access to authorized users. It reduces the complexity and cost of security administration in large networked applications by creating roles for various job functions. Each role specifies the permission to perform a certain set of operations. Users are assigned these roles that allow them to acquire permission to perform certain functions.

Because users are not assigned permissions directly but only acquire them through their role (or roles), management of individual user rights occurs by assigning the appropriate roles to the user, which simplifies such common operations as adding a user or changing a user's department.

createComponentGroups

Creates one or more component group objects in a network element. Given the InstanceNameId of a network element and a list of ComponentGroup objects, creates the objects in the server and returns its InstanceNameIds.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid InstanceNameId of a network element.
- If the componentGroupCol is null or the collection is empty.
- If the componentGroupCol contains one or more null element, or the collection contains objects that are not of type ComponentGroup.

PropertiesException is thrown if any of the following situation occurs:

- In the componentGroupCol collection, if any of the ComponentGroup attribute is not valid.

Example:

- name of a ComponentGroup starts with a question mark. Because, component group name cannot contain a space or question mark.

IntegrityException is thrown if any of the following situation occurs:

- If the componentGroupCol contains a ComponentGroup that already exist in the database.
createNetworkElementUserRoles

Creates one or more network element user role objects in a network element. Given the InstanceNameId of a network element and a list of user role objects, creates the objects in the server and returns its instance name IDs.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the networkElementUserRoleCol is null or the collection is empty.
- If the networkElementUserRoleCol contains one or more null element, or the collection contains objects that are not of type NetworkElementUserRole.

PropertiesException is thrown if any of the following situation occurs:

- In the networkElementUserRoleCol collection, if any of the NetworkElementUserRole attribute is not valid.

IntegrityException is thrown if any of the following situation occurs:

- If the networkElementUserRoleCol contains a NetworkElementUserRole that already exist in the database.

This API will not consider the user association. If a user role is passed with the user association, that will not be considered by this API. User needs to call separate API to bind the user to a user role.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of a network element.
- networkElementUserRoleCol—a collection (one or more) of NetworkElementUserRole objects that needs to be created.

Return Value

Instance name IDs of the newly created NetworkElementUserRole objects.

createNetworkElementUsers

Creates one or more network element user objects in a network element. Given the InstanceNameId of a network element and a list of network element user objects, creates the objects in the server and returns its instance name IDs.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of a network element.
- networkElementUserCol—a collection (one or more) of NetworkElementUser objects that needs to be created.

Return Value

Instance name IDs of the newly created NetworkElementUser objects.
deleteComponentGroups

Deletes one or more component group objects. Given the InstanceNameId of the ComponentGroup objects, those objects will be deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If componentGroupNameCol collection is null or it is empty.
- If componentGroupNameCol collection contains an element that is not of type ComponentGroupNameCol.
- If componentGroupNameCol collection contains a ComponentGroup that does not exist in the database.

Parameters

- opContext—Operational context
- componentGroupNameCol—a collection that contains InstanceNameId of one or more ComponentGroup objects that needs to be deleted.

Return Value

void
deleteNetworkElementUserRoles

Deletes one or more network element user role objects. Given the InstanceNameId of the NetworkElementUserRole objects, those objects will be deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If networkElementUserRoleInstanceNameIdCol collection is null or it is empty.
- If networkElementUserRoleInstanceNameIdCol collection contains an element that is not of type NetworkElementUserRole InstanceNameId.
- If networkElementUserRoleInstanceNameIdCol collection contains a NetworkElementUserRole that does not exist in the database.

Parameters

opContext—Operational context

networkElementUserRoleInstanceNameIdCol—a collection that contains InstanceNameId of one or more NetworkElementUserRole objects that needs to be deleted.

Return Value

void

deleteNetworkElementUsers

Deletes one or more network element user objects. Given the InstanceNameId of the NetworkElementUser objects, those objects will be deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If networkElementUserInstanceNameIdCol collection is null or it is empty.
- If networkElementUserInstanceNameIdCol collection contains an element that is not of type NetworkElementUser InstanceNameId.
- If networkElementUserInstanceNameIdCol collection contains a NetworkElementUser that does not exist in the database.

Parameters

opContext—Operational context

networkElementUserInstanceNameIdCol—a collection that contains InstanceNameId of one or more NetworkElementUser objects that needs to be deleted.

Return Value

void

getComponentAuths

Returns ComponentAuth objects from it's InstanceNameIds. Given a collection of InstanceNameId of ComponentAuth, returns corresponding ComponentAuth objects.

ValidationException is thrown if any of the following situation occurs:
getComponentAuthsInNetworkElement

Returns all the components configured in a network element. Given the InstanceNameId of the network element, returns a collection of component objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
- opContext—Operational context
- neInstanceNameId—InstanceNameId of the network element.

Return Value
List of ComponentAuth objects. In the returned list of ComponentAuth objects, only the CommandAuth objects associated with returned ComponentAuth objects will be present, and all other associations will be cleared.

gGetComponentGroups

Returns ComponentGroup objects from it's InstanceNameIds. Given a collection of InstanceNameId of ComponentGroup, returns corresponding ComponentGroup objects.

ValidationException is thrown if any of the following situation occurs:
- If componentGroupInstanceNameIdCol is null or it is empty.
- If componentGroupInstanceNameIdCol contains invalid InstanceNameId of a ComponentGroup.
- If componentGroupInstanceNameIdCol collection contains a null value.
- If there is no equivalent ComponentGroup object with the given InstanceNameId in the componentGroupInstanceNameIdCol.

Parameters
- opContext—Operational context
componentGroupNameInstanceNameIdCol—a collection of InstanceNameId of ComponentGroup.

Return Value
List of ComponentGroup objects corresponding to given collection of InstanceNameId. In the returned list of ComponentGroup objects, only the ComponentGroupAuthRule objects and the ComponentAuth objects associated with returned ComponentGroup objects will be present, and all other associations will be cleared. The ComponentAuth objects will contain the CommandAuth objects associated to it.

gGetComponentGroupsInNetworkElement

Returns all component groups configured in a network element. Given the InstanceNameId of the network element, returns a collection of component group objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId of the network element.

Return Value
List of ComponentGroup objects. In the returned list of ComponentGroup objects, only the ComponentGroupAuthRule objects and the ComponentAuth objects associated with returned ComponentGroup objects will be present, and all other associations will be cleared. The ComponentAuth objects will contain the CommandAuth objects associated to it.

gGetNetworkElementUserRoles

Returns NetworkElementUserRole objects from it's InstanceNameIds. Given a collection of InstanceNameId of NetworkElementUserRole, returns corresponding NetworkElementUserRole objects.

ValidationException is thrown if any of the following situation occurs:

- If networkElementUserRoleInstanceNameIdCol is null or it is empty.
- If networkElementUserRoleInstanceNameIdCol contains invalid InstanceNameId of a NetworkElementUserRole.
- If networkElementUserRoleInstanceNameIdCol contains a null value.
- If there is no equivalent NetworkElementUserRole object with the given InstanceNameId in the networkElementUserRoleInstanceNameIdCol.

Parameters
opContext—Operational context
networkElementUserRoleInstanceNameIdCol—a collection of InstanceNameId of NetworkElementUserRole.
Return Value

List of NetworkElementUserRole objects corresponding to given collection of InstanceNameId. In the returned list of NetworkElementUserRole objects, only the following associations will be present, and all other associations will be cleared.

- All associated RoleBasedAuthorizationRule objects of the returned NetworkElementUserRole object. If the RoleBasedAuthorizationRule is an instance of ComponentGroupAuthRule, then associated ComponentGroup will also be returned along with the ComponentAuth objects. If the RoleBasedAuthorizationRule is an instance of ComponentAuthRule, then associated ComponentGroup will also be returned along with the ComponentAuth objects. CommandAuth objects associated to ComponentAuth objects will also be returned.
- VlanPolicy association, if any, in NetworkElementUserRole.
- VsanPolicy association, if any, in NetworkElementUserRole.
- VrfPolicy association, if any, in NetworkElementUserRole. Except Vrf objects, if the VlanPolicy has any other associations, those associations will be cleared.
- NetworkInterfacePolicy association, if any, in NetworkElementUserRole. Except NetworkInterface, if the NetworkInterfacePolicy has any other associations, those associations will be cleared.

getNetworkElementUserRolesInNetworkElement

Returns all network element user role configured in a network element. Given the InstanceNameId of the network element, returns a collection of user role objects.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.

Parameters

opContext—Operational context

neInstanceNameId—InstanceNameId of the network element.

Return Value

List of NetworkElementUserRole objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.

- All associated RoleBasedAuthorizationRule objects of the returned NetworkElementUserRole object. If the RoleBasedAuthorizationRule is an instance of ComponentGroupAuthRule, then associated ComponentGroup will also be returned along with the ComponentAuth objects. If the RoleBasedAuthorizationRule is an instance of ComponentAuthRule, then associated ComponentGroup will also be returned along with the ComponentAuth objects. CommandAuth objects associated to ComponentAuth objects will also be returned.
- VlanPolicy association, if any, in NetworkElementUserRole.
- VsanPolicy association, if any, in NetworkElementUserRole.
- VrfPolicy association, if any, in NetworkElementUserRole. Except Vrf objects, if the VlanPolicy has any other associations, those associations will be cleared.
- NetworkInterfacePolicy association, if any, in NetworkElementUserRole. Except NetworkInterface, if the NetworkInterfacePolicy has any other associations, those associations will be cleared.
getNetworkElementUsers

Returns NetworkElementUser objects from it's InstanceNameIds. Given a collection of InstanceNameId of NetworkElementUser, returns corresponding NetworkElementUser objects.

ValidationException is thrown if any of the following situation occurs:

- If networkElementUserInstanceNameIdCol is null or it is empty.
- If networkElementUserInstanceNameIdCol contains invalid InstanceNameId of a NetworkElementUser.
- If networkElementUserInstanceNameIdCol contains a null value.
- If there is no equivalent NetworkElementUser object with the given InstanceNameId in the networkElementUserInstanceNameIdCol.

Parameters

opContext—Operational context

networkElementUserInstanceNameIdCol—a collection of InstanceNameId of NetworkElementUser.

Return Value

List of NetworkElementUser objects corresponding to given collection of InstanceNameId. In the returned list of NetworkElementUser objects, only the following associations will be present, and all other associations will be cleared.

- Associated NetworkElementUserCredentials object, if any.
- NetworkElementUserRole association for every NetworkElementUser, if any. If that NetworkElementUserRole has other associations like RoleBasedAuthorizationRule entries and so on, those associations will be cleared.

getNetworkElementUsersInNetworkElement

Returns all NetworkElementUser objects in a network element. Given the InstanceNameId of a network element, returns a collection of NetworkElementUser objects in the network element.

ValidationException is thrown if the argument passed is null or it is not a valid InstanceNameId of an AbstractNetworkElement.

Parameters

opContext—Operational context

neInstanceNameId—InstanceNameId of the network element.

Return Value

List of NetworkElementUser objects. In the returned list of objects, only the following associations will be preserved, and all other associations will be cleared.

- Associated NetworkElementUserCredentials object, if any.
- NetworkElementUserRole association for every NetworkElementUser, if any. If that NetworkElementUserRole has other associations like RoleBasedAuthorizationRule entries and so on, those associations will be cleared.
modifyComponentGroups

Modifies one or more existing ComponentGroup objects.

ValidationException is thrown if any of the following situation occurs:
- If componentGroupCol collection is null or it is empty.
- If componentGroupCol collection contains an object that is not of type ComponentGroup.

PropertiesException is thrown if any of the following situation occurs:
- In the componentGroupCol collection, if any attribute in the ComponentGroup is not valid.

IntegrityException is thrown if any of the following situation occurs:
- If the componentGroupCol collection contains a ComponentGroup that does not exist in the database.

Parameters

- opContext—Operational context
- componentGroupCol—a collection (one or more) of ComponentGroup objects that will replace the existing ComponentGroup objects in the database.

Return Value

void

modifyNetworkElementUserRoles

Modifies one or more existing network element user role objects.

ValidationException is thrown if any of the following situation occurs:
- If networkElementUserRoleCol collection is null or it is empty.
- If networkElementUserRoleCol collection contains an object that is not of type NetworkElementUserRole.

PropertiesException is thrown if any of the following situation occurs:
- In the networkElementUserRoleCol collection, if any attribute in the NetworkElementUserRole is not valid.

IntegrityException is thrown if any of the following situation occurs:
- If the networkElementUserRoleCol collection contains a NetworkElementUserRole that does not exist in the database.

This API will not consider the user association. If a user role is passed with the user association, that will not be considered by this API. User needs to call separate API to bind the user to a user role.

Parameters

- opContext—Operational context
- networkElementUserRoleCol—a collection (one or more) of NetworkElementUserRole objects that will replace the existing NetworkElementUserRole objects in the database.
modifyNetworkElementUsers

Modifies one or more existing network element user objects.

ValidationException is thrown if any of the following situation occurs:
- If networkElementUserCol collection is null or it is empty.
- If networkElementUserCol collection contains an object that is not of type NetworkElementUser.

PropertiesException is thrown if any of the following situation occurs:
- In the networkElementUserCol collection, if any attribute in the NetworkElementUser.

IntegrityException is thrown if any of the following situation occurs:
- If the networkElementUserCol collection contains a NetworkElementUser that does not exist in the database.

Actual Password will not be stored in the database. Instead a dummy value will be stored.

Parameters
opContext—Operational context
networkElementUserCol—a collection (one or more) of NetworkElementUser objects that will replace the existing NetworkElementUser objects in the database.

Return Value
void

Return Value
List of NetworkElementUserRole objects. In the returned list of objects, only the following associations will be present, and all other associations will be cleared.
- All associated RoleBasedAuthorizationRule objects of the returned NetworkElementUserRole object. If the RoleBasedAuthorizationRule is an instance of ComponentGroupAuthRule, then associated ComponentGroup will also be returned.
- VlanPolicy association, if any, in NetworkElementUserRole.
- VsanPolicy association, if any, in NetworkElementUserRole.
- VrfPolicy association, if any, in NetworkElementUserRole. Except Vrf objects, if the VlanPolicy has any other associations, those associations will be cleared.
- NetworkInterfacePolicy association, if any, in NetworkElementUserRole. Except NetworkInterface, if the NetworkInterfacePolicy has any other associations, those associations will be cleared.
RepositoryApp Service

This chapter describes the DCNM web services’ API methods for the RepositoryApp service.

Information About RepositoryApp Service

This chapter defines the services exposed by the Repository API component of the DCNM server.

abortCurrentProcess

This API will be usually called after invoking the API getResultsFromConnection(OpContext, String) to stop the current process running. Client can also invoke this API if the End of list is not reached after invoking getFirstLevelFileNames(OpContext, String, String, String, Boolean), getRecursivelyFileNames(OpContext, String, String, String, Boolean). This API will return a boolean value = true for successful abort.

Parameters
opContext—Operational context
collectionId—ConnectionId for which the current process has to be aborted

Return Value
True for successful abort, else false.

createConnection

This API is used by the client to create a connection to the repository which is in the DB. Creates a connection to the give repository InstanceNameId. Generates a unique connection Id string. This is unique across each client connections, client can create multiple connection to a single repository server with unique connection Id. Returns this connectionId to the client.

Parameters
opContext—Operational context
repositoryInstanceNameId—Repository Instance name for which a connection has to be created
createConnectionWithRepSvr

This API is used by the client to create a temporary connection by giving a dummy ServerRepository object. This API is usually called when client wants to connect to a repository which is not created earlier. Hence the client can create a connection using this API. Creates a connection to the repository in the given ServerRepository(rfp) information. Generates a unique connection Id string. This is unique across each client connections, client can create multiple connection to a single repository server with unique connection Id. Returns this connectionId to the client.

Parameters
- opContext—Operational context
- rfp—Information of the repository for which a connection has to be created

Return Value
Returns a connection Id to the client. Client can use the same connection Id in successive Repository API calls on the same repository instead of new connection.

decreaseRepositoryServers

Persist the repository information collection in DCNM server and returns the repos itory server collection itself. Since few properties will be manipulated and set in the server side. Client may require to know it.

Parameters
- opContext—Operational context.
- newRepLocationCol—Collection of Repository information to persist,

Return Value
Returns the repository server collection to the client.

deleteConnection

Disconnect and remove the repository connection for the given connectionId from the DCNM server.

Parameters
- opContext—Operational context
- connectionId—ConnectionId for which the connection should be disconnected and remove from DCNM server.

Return Value
void
deleteConnectionUseRepSvr

This API is used by the client, if he wants the DCNM server to disconnect and remove all connections from the given set of server repositories. This will be likely used before the ServerRepository information is going to be modified. Disconnect and remove the repository connections for the given repository information from the DCNM server.

Parameters

opContext—Operational context
repositoryCol—Repository information collection for which the connection has to be created and disconnected.

Return Value

void

deleteRepositoryFiles

This API deletes the given files on the given repository connection.

Parameters

opContext—Operational context
connectionId—connection id of a repository
pathCol—File's path collection which are to be deleted on the repository(which is containing connection connectionId)

Return Value

void

deleteRepositoryServers

Removes the repository informations from the DCNM server for the given Repository instanceNameId collection.

Parameters

opContext—Operational context.
repInstanceNameIdCol—Repository instance name collection for which repository information to be removed.

Return Value

void

getAllRepositoryServers

Retrieves all the repository information to the client.
getFirstLevelFileNames

Returns the immediate file/directory names, its details under the given directory alone and filter is applied. For the retrieved file and directory names & details a List object is populated and returned.

If the given directory name is null, the retrieval of files names will be done from the repository's root directory.

If the given filter is null, all the retrieved file names are sent to client.

If the given filter is not null, the directory/file names are verified against the filter, if the verification is false, the directory/file names will be skipped. This API waits for few seconds for the result otherwise it returns the result got at that time. This API creates a background process for further result. Client can invoke getResultsFromConnection(OpContext, String) API for further results. The end of the result can be identified by a file which contains the name ":End:".

Parameters
opContext—Operational context
connectionId—ConnectionId for which the retrieval to be done from which repository
dir—dir is the directory, under which the file/directory names retrieval will be done.
filter—filter is applied on the retrieved directory and file names from repository
includeFileNames—Whether file names also will be retrieved or only directory names from repository

Return Value
Returns the List which is retrieved and populated from the repository for the given connectionId.

gerRecursivelyFileNames

Retrieve the file names, its details recursively from the given directory and filter is applied. For the retrieved file and directories details a List object is populated and returned.

If the given directory detail is null, the retrieval will be done from the repository's root directory.

If the given filter is null, all the retrieved file details are sent to client.

If the given filter is not null, the directory/file names are verified against the filter, if the verification is false, the directory/file information is skipped. This API waits for few seconds for the result otherwise it returns the result got at that time. This API creates a background process for further result. Client can invoke getResultsFromConnection(OpContext, String) API for further results. The end of the result can be identified by a file which contains the name ":End:".

Parameters
opContext—Operational context
connectionId—ConnectionId for which the retrieval to be done from which repository
getResultsFromConnection

This API returns the accumulated result of the current process running or for the current connection. Client usually invoke this API if the End of list is not reached after invoking getFirstLevelFileNames(OpContext, String, String, String, Boolean), getRecursivelyFileNames(OpContext, String, String, String, Boolean) This API will be invoked from a loop to get the result of the current process till End of list is reached. The End of List can be identified from a file which contains the name ':End:' in this result collection. This returns the result from the process and clears the same from it, since the successive invocation of this API wont return again the same result collection. The current process can be aborted by the API abortCurrentProcess(OpContext, String)

Parameters
opContext—Operational context
connectionId—ConnectionId for which the retrieval to be done from which repository

Return Value
Returns the List which is retrieved and populated from the repository for the given connectionId.

modifyRepositoryServers

Modifies the given repository information collections in the DCNM server and returns the same collection.

Parameters
opContext—Operational context
modRepLocationCol—Modified repository information collection.

Return Value
Returns the modified repository information collection.

validateConnectivity

This API is used to verify whether the Repository is able to connect and returns the appropriate message of the cause for connection failure.

Parameters
opContext—Operational context
repositoryCol—repository collection for which the connection has to be validated

Return Value
Returns the pair object collection, where a pair object holds the reference to the boolean attribute and a message string. Boolean attribute refers to the repository is able to get connected. Message String contains the information cause of the connection failure.
SecurityApp Service

This chapter describes the DCNM web services’ API methods for the SecurityApp service.

Information About SecurityApp Service

Security services provide the various functions related to DCNM server access control. Access control allows you to control who is allowed access to DCNM server resources and what services they are allowed to access. Typically, users are assigned an ID number (session ID) and password that allows them to access information and to perform operations within their authority. The API categories are as follows:

- **Query/Get APIs**—Query security related information (authorized operations) from the persistent database.
- **Create APIs**—Create new application user.
- **Modify APIs**—Modify basic attributes of existing application user.
- **Delete APIs**—Delete existing application user.

**getSessionId**

This operation creates and returns the session Id for the current user. The session Id is used for subsequent requests.

**Parameters**

opContext—the operational context.

**Return Value**

Session ID String.
getSessionId
ServerAdminApp Service

This chapter describes the DCNM web services’ API methods for the ServerAdminApp service.

Information About ServerAdminApp Service

This chapter defines the APIs exposed by the Server Administration service feature.

getServerSetting

Gets the setting of the DCNM server.

Parameters
opContext—Operational context

Return Value
ServerSetting

getServerStatus

Gets the status of the DCNM server.

Parameters
opContext—Operational context

Return Value
ServerStatus

modifyServerSetting

Modifies the DCNM server setting.
modifyServerSetting

Parameters
- opContext—Operational context
- ServerSetting—server setting

Return Value
- ServerSetting
SoftwareApp Service

This chapter describes the DCNM web services’ API methods for the SoftwareApp service.

Information About SoftwareApp Service

This chapter defines the APIs that you can use to get information about the installed software on the DCNM server.

getProduct

Returns information about the DCNM product.

Return Value
DCNM product information.
SpanApp Service

This chapter describes the DCNM web services’ API methods for the SpanApp service.

Information About SpanApp Service

Switches use the switch port analyzer (SPAN) feature to copy traffic from switch source ports or VLANs to destination ports. This feature allows you to analyze traffic at destination ports by using monitoring devices such as network analyzers. The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Modify APIs—Modify configurations of SPAN sessions or global settings.
- Delete APIs—Delete SPAN sessions from the device.

The following sessions are supported for query APIs and delete APIs:

- Local SPAN
- RSPAN source
- RSPAN destination

You can modify only local SPAN sessions. An exception is thrown for RSPAN source or destination sessions.

The following associations are available for interfaces:

- Interface name
- Interface status

The following associations are available for VLANs:

- VLAN status

addDestinationPortsToSpanSession

Adds destination ports/Ethernet channels to local SPAN session. Remote SPAN is not supported.

ValidationException is thrown if any of the following situation occurs:

- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If NetworkInterface which is already used as SPAN source is used as destination
If the NetworkInterface is already configured as SPAN destination for any other SPAN session.

If the SPAN permit-list is enabled and NetworkInterface is not configured as permit-list destination port.

If SwitchedEtherChannelNetworkInterface / RoutedEtherChannelNetworkInterface is used as SPAN destination in case of Catalyst 6500 series switches

If Etherchannel member ports are configured as SPAN destination

If the NetworkInterface is not SPAN destination capable.

If any of the objects corresponding to InstanceNameId in dstPortNameIds does not exist

Parameters

- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- dstPortNameIds—List of InstanceNameId of the interfaces to be added as SPAN destination.

Return Value

void

addFilteredVlanRangeToSpanSession

Adds filtered VLANs to local SPAN session. Remote SPAN is not supported. Can add a VLAN that does not existing in the device (pre-provisioning).

ValidationException is thrown if any of the following situation occurs:

- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If adding filtered VLANs to the local SPAN and if the given local SPAN session has source VLANs added.

Parameters

- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- filteredVlans—range of VLAN IDs to be configured as filtered VLANs. Eg: "2,7-9".

Return Value

void

addFilteredVlansToSpanSession

Adds filtered VLANs to local SPAN session. Remote SPAN is not supported. Can add a VLAN which is not existing in the device as well (preprovisioning).

ValidationException is thrown if any of the following situation occurs:

- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist

Parameters

- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- filteredVlans—range of VLAN IDs to be configured as filtered VLANs. Eg: "2,7-9".

Return Value

void
addPortsToSpanPermitList

Adds new ports to SPAN destination permit list.

ValidationException is thrown if any of the following situation occurs:

- If neNameId is null or not InstanceNameId of the AbstractNetworkElement.
- If the specified AbstractNetworkElement does not exist
- If any of the element's InstanceNameId in interfaceNameIds is of type RoutedEtherChannelNetworkInterface or SwitchedEtherChannelNetworkInterface.

Parameters

- opContext—Operational context
- neNameId—InstanceNameId of the network element.
- interfaceNameIds—List of InstanceNameId of NetworkInterface

Return Value

void

addSourcesToSpanSession

Adds source ports/Ethernet channels to a local SPAN session. Remote SPAN is not supported.

ValidationException is thrown if any of the following situation occurs:

- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If NetworkInterface which is already used as destination is used as SPAN source
- If SwitchedEtherChannelNetworkInterface / RoutedEtherChannelNetworkInterface is used as SPAN source in case of Catalyst 6500 series switches
- If Etherchannel member ports are configured as SPAN source
- If NetworkInterface is added as SPAN source if the session already has filtered VLANs configured in case of Catalyst 6500 series switches
- If NetworkInterface is added as SPAN source if the session already has VLANs configured as SPAN source in case of Catalyst 6500 series switches

Parameters

- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- vlanRange—range of VLAN IDs to be configured as filtered VLANs. Eg: "2,7-9".

Return Value

void
addSourcesToSpanSessionByRange

Adds source VLANs to a local SPAN session. Remote SPAN is not supported.

ValidationException is thrown if any of the following situation occurs:

- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If VLAN is added as SPAN source if the session already has filtered VLANs configured in case of Catalyst 6500 series switches
- If VLAN is added as SPAN source if the session already has Interfaces configured as SPAN source in case of Catalyst 6500 series switches
- If any of the objects corresponding to VLAN ID in vlanRange does not exist
- If direction is null

Parameters

- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- srcNameIds—List of InstanceNameId of NetworkInterface / VLAN to be configured as sources for the given SPAN session.
- direction—Traffic direction SpanSourceDirection.

Return Value

void

createSpanSessions

Create a SPAN sessions in a network element. This method can be used to create all types of SPAN sessions.
While calling this method, one has to set the NetworkElement reference to the session on which the corresponding session has to be created.

On the server, this method will create the new InstanceNameId for the SPAN session based on the session ID and NetworkElement.

For creating a SPAN session, the user must configure at least one source or destination association to it.

Following associations will be updated:

- SpanSource Local SPAN Source—Source interfaces and VLANs
- VLAN Local SPAN Source—Filtered VLANs
- NetworkInterface Local SPAN Destination—Destination interfaces
- SPAN status

ValidationException is thrown if any of the following situation occurs:

- RSPAN source/destination sessions are to be created
- If spanSessionList is null or empty or elements in the list are not of type LocalSpanSession.
- If the LocalSpanSession object in the spanSessionList does not have valid session ID or null AbstractNetworkElement reference.
- If the number of SPAN sessions limit in the specified AbstractNetworkElement exceeded
- If the SPAN session to be created does not have source or destination associations
- In Catalyst 6500 series switches, if the SPAN session to be created have NetworkInterface and VLAN as SPAN sources
- If the SPAN session to be created have NetworkInterface as SPAN destination interface which is already a SPAN destination some other session

PropertiesException is thrown if any of the following situation occurs:

- In the spanSessionList collection, if any one of the SpanSession attribute is not valid.

Example:

- sessionId of a SpanSession is out of range.

IntegrityException is thrown if any of the following situation occurs:

- In the spanSessionList collection, if any one of the SpanSession object is already existing in the specified AbstractNetworkElement.

**Parameters**

opContext—Operational context

spanSessionList—List of (@ link SpanSession)Objects.

**Return Value**

List of InstanceNameId for the newly created SPAN sessions.

### createSpanSessionsInNetworkElement

Create a list of SPAN sessions in a network element. This method can be used to create any type of SPAN session. InstanceNameId will be created on the server based on the session ID and the NetworkElement InstanceNameId.
Following associations will be updated:

- SpanSource Local SPAN Source—Source interfaces and VLANs
- VLAN Local SPAN Source—Filtered VLANs
- NetworkInterfaceLocal SPAN Destination—Destination interfaces
- SPAN status

ValidationException is thrown if any of the following situation occurs:

- If sessionInstanceNameIds is not a valid InstanceNameId of SpanSession object.
- If neNameId is null or it is not of InstanceNameId of AbstractNetworkElement.

ValidationException is thrown if any of the following situation occurs:

- RSPAN source/destination sessions are to be created
- If spanSessionList is null or empty or elements in the list are not of type LocalSpanSession.
- If the specified AbstractNetworkElement does not exist
- If the LocalSpanSession object in the spanSessionList does not have valid session ID or null AbstractNetworkElement reference.
- If the number of SPAN sessions limit in the specified AbstractNetworkElement exceeded
- If the SPAN session to be created does not have source or destination associations
- In Catalyst 6500 series switches, If the SPAN session to be created have NetworkInterface and VLAN as SPAN sources
- If the SPAN session to be created have NetworkInterface as SPAN destination interface which is already a SPAN destination some other session

PropertiesException is thrown if any of the following situation occurs:

- In the spanSessionList collection, if any one of the SpanSession attribute is not valid.

Example:

- sessionId of a SpanSession is out of range.

IntegrityException is thrown if any of the following situation occurs:

- In the spanSessionList collection, if any one of the SpanSession object is already existing in the specified AbstractNetworkElement.

**Parameters**

- opContext—Operational context
- neNameId—InstanceNameId of the network element.
- spanSessionList—List of SpanSession that need to be created.

**Return Value**

Ordered list of newly created SPAN sessions instance name IDs

**deleteSpanSessions**

Deletes the SPAN session corresponding to the given list of InstanceNameId. The corresponding SpanSession and its associations will be deleted.
The following sessions are allowed to be deleted:

- Local SPAN
- RSPAN Source
- RSPAN Destination

ValidationException is thrown if any of the following situation occurs:

- If sessionNameIds is null or empty or elements in the list are not InstanceNameId of SpanSession.
- In the sessionNameIds collection, if any one of the SpanSession object corresponding to InstanceNameId in the sessionNameIds does not exist.

**Parameters**

- **opContext**—Operational context
- **sessionNameIds**—List of SPAN session InstanceNameId that need to be deleted.

**Return Value**

void

---

**disableInterfacesMonitor**

Disables Interfaces Monitor for one or more Interfaces. Given the instance name ID of interface, InterfacesMonitor will be disabled on those elements. This API is to disable "InterfacesMonitor" for Cisco NX-OS devices and it is not applicable for other Cisco IOS devices.

ValidationException is thrown if any of the following situation occurs:

- If the interfaceInstanceNameId is of type NetworkInterface null.
- If the interfaceInstanceNameId contains one or more null element, or the collection is empty or it is not type InstanceNameId.
- If the given interface is running in other Cisco IOS

**Parameters**

- **opContext**—Operational context
- **interfaceInstanceNameId**—List of InstanceNameId of NetworkInterface for which the information is required

**Return Value**

void

---

**enableInterfacesMonitor**

Enables Interface Monitor for the given Interfaces. Given the instance name ID of one or more interfaces, "InterfacesMonitor" will be enabled on those elements. This API is to enable InterfacesMonitor for Cisco NX-OS devices and it is not applicable for other Cisco IOS devices.

ValidationException is thrown if any of the following situation occurs:

- If the interfaceInstanceNameId is null.
getLocalSpanSessionsInNetwork

Returns local SPAN sessions corresponding to a list of session IDs for a list of network elements. Following associations will be available:
- Source interfaces and VLANs
- Filtered VLANs
- Destination interfaces
- SPAN status

ValidationException is thrown if any of the following situation occurs:
- If elements in the neNameIdList are not of InstanceNameId type of AbstractNetworkElement.

Parameters
- opContext—Operational context
- sessionId—range of session IDs (e.g. 1-2,20,50-53). If null, returns all sessions.
- neNameIdList—List of InstanceNameId of network elements whose Session ID are to be queried. If null or empty, will return the sessions from all the managed network elements.

Return Value
Ordered list of Local SPAN sessions

getSpanDestinationPorts

Returns the destination ports for the given session NameId.
Following associations will be available for resultant Interfaces:
- Interface Name
- Interface Status

ValidationException is thrown if any of the following situation occurs:
- If sessionNameId is null or not InstanceNameId of the SpanSession.
- If the specified SpanSession does not exist

Parameters
- sessionNameId—InstanceNameId of the SpanSession
- neNameIdList—List of InstanceNameId of network elements whose Session ID are to be queried. If null or empty, will return the sessions from all the managed network elements.

Return Value
Ordered list of Local SPAN sessions
getSpanPermitListSettingsInNetwork

Returns SPAN destination permit list for a list of network elements.
Following associations will be available:
• NetworkInterface Destination interfaces

ValidationException is thrown if any of the following situation occurs:
• If elements in the neNameIdList are not of InstanceNameId type of AbstractNetworkElement.

Parameters
opContext—
neNameIdList—List of InstanceNameId of network elements whose SpanPermitListSetting are to be queried. If null or empty, will return the sessions from all the managed network elements.

Return Value
List of SpanPermitListSetting with its associations initialized.

getSpanSessions

Returns list of SPAN sessions with all its associations for a given list of instance name IDs of the SPAN sessions.
The return type can have the following session types:
• Local SPAN
• RSPAN Source
• RSPAN Destination

Following associations will be available:
• SpanSource Local SPAN/RSPAN Source—Source interfaces and VLANs
• VLAN Local SPAN/RSPAN Source—Filtered VLANs
• NetworkInterface Local SPAN/RSPAN Destination—Destination interfaces
• VlanRSPAN Source/RSPAN Destination VLAN—RSPAN VLAN
• SPAN status

ValidationException is thrown if any of the following situation occurs:
• If sessionInstanceNameIds is null or empty or it is not of type InstanceNameId.
• If sessionInstanceNameIds is not a valid InstanceNameId of SpanSession object.
getSpanSessionsAssociatedToVlan

Returns SPAN session objects for which given VLAN is configured as SPAN source.

Following associations will be available for each SPAN session:

25. Local SPAN to VLAN association

Following associations will be available on other end of LocalSpanSessionAppliesToVlan:

26. Reference to VLAN

ValidationException is thrown if any of the following situation occurs:

- If vlanNameId is null or not InstanceNameId of the VLAN.
- If the specified VLAN does not exist

Parameters

- opContext—Operational context
- vlanNameId—InstanceNameId of the VLAN.

Return Value

List of SpanSession objects for which given VLAN is configured as SPAN source.

getSpanSessionsInNetwork

Returns SPAN sessions corresponding to a list of session IDs for a list of network elements.

The return type can have the following session types:

- Local SPAN
- RSPAN Source
- RSPAN Destination

Following associations will be available:

- SpanSource Local SPAN/RSPAN Source—Source interfaces and VLANs
- VLAN Local SPAN/RSPAN Source—Filtered VLANs
- NetworkInterface Local SPAN/RSPAN Destination—Destination interfaces
- VLAN RSPAN Source/RSPAN Destination VLAN—RSPAN VLAN
- SPAN status

ValidationException is thrown if any of the following situation occurs:

- If elements in the neNameIdList are not of InstanceNameId type of AbstractNetworkElement.
getSpanSources

Returns all source ports/etherchannels/VLANs associated to a local SPAN session or remote SPAN source session. The return type will have the list of following objects for Local SPAN session:

19. Interface associations
20. VLAN associations

Following associations will be available for LocalSpanSessionAppliesToNetworkInterface:
21. Reference to interface
Following associations will be available for LocalSpanSessionAppliesToVlan:
22. Reference to VLAN

ValidationException is thrown if any of the following situation occurs:
- If sessionNameId is null or not InstanceNameId of the SpanSession.
- If the specified SpanSession does not exist

Parameters
opContext—Operational context
sessionNameId—InstanceNameId of a SPAN session

Return Value
List of sources for a given SPAN session

modifyFilteredVlanRangeForSpanSession

Modifies the filtered VLANs associated to local SPAN session. The existing filtered VLANs will be replaced with the given VLANs. Remote SPAN is not supported.

ValidationException is thrown if any of the following situation occurs:
- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If any of the objects corresponding to InstanceNameId in vlanInstanceNameIds does not exist
- If adding filtered VLANs to the local SPAN and if the given local SPAN session has source VLANs added.

Parameters
opContext—Operational context
sessionNameId—InstanceNameId of a SPAN session

Return Value
List of sources for a given SPAN session
Send document comments to nexus7k-docfeedback@cisco.com

modifyFilteredVlansByRangeForSpanSession

Parameters
- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- filteredVlans—range of VLAN IDs to be configured as filtered VLANs. Eg:”2,7-9”.

Return Value
void

modifyFilteredVlansByRangeForSpanSession

Modifies the filtered VLANs associated to local SPAN session. The existing filtered VLANs will be replaced with the given VLANs. Remote SPAN is not supported.

ValidationException is thrown if any of the following situation occurs:
- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If adding filtered VLANs to the local SPAN and if the given local SPAN session has source VLANs added.

Parameters
- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- vlanRange—range of VLAN IDs to be configured as filtered VLANs. Eg:”2,7-9”.

Return Value
void

modifyFilteredVlansForSpanSession

Modifies the filtered VLANs associated to local SPAN session. The existing filtered VLANs will be replaced with the given VLANs. Remote SPAN is not supported.

ValidationException is thrown if any of the following situation occurs:
- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If any of the objects corresponding to InstanceNameId in vlanInstanceNameIds does not exist
- If adding filtered VLANs to the local SPAN and if the given local SPAN session has source VLANs added.

Parameters
- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- vlanInstanceNameIds—List of InstanceNameId of VLAN to be replaced as filtered VLANs to the given SPAN session.
modifySpanPermitListSettingsInNetworkElement

Modify the permit list for a given network element. This method include modification to the destination ports that are associated to SpanPermitListSetting.

Following associations will be updated:
- NetworkInterface destination interfaces associated to permit-list

ValidationException is thrown if any of the following situation occurs:
- If neNameId is null or not InstanceNameId of the AbstractNetworkElement.
- If the specified AbstractNetworkElement does not exist
- If modifiedPermitListSetting is null.
- If permitlist is Enabled without associating a port as permit-list destination port.
- If permitlist is Enabled without associating a port as permit-list destination port.
- If the NetworkInterface configured is of type RoutedEtherChannelNetworkInterface or SwitchedEtherChannelNetworkInterface.

Parameters
opContext—Operational context
neNameId—InstanceNameId of the AbstractNetworkElement.
modifiedPermitListSetting—Modified SpanPermitListSetting object.

Return Value
void

modifySpanSessions

Modify existing SPAN sessions. This method will not modify the associations corresponding to the SPAN session.

ValidationException is thrown if any of the following situation occurs:
- If modifiedSpanSessions is null or empty or elements in the list are not of type SpanSession.

PropertiesException is thrown if any of the following situation occurs:
- If any of the attributes for SpanSession object modifiedSpanSessions is not valid

Example:
- description of a SpanSession is only applicable to Cisco NX-OS platform.
- sessionId of a SpanSession is out of range.

IntegrityException is thrown if any of the following situation occurs:
- If anyone of the object in modifiedSpanSessions collection SpanSession object does not exist.
modifySpanSourceTrafficDirection

Modifies the direction of the traffic flowing from a span source port/EtherChannel or VLAN. Remote SPAN is not supported.

ValidationException is thrown if any of the following situation occurs:

- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If srcNameId is null or not
- If the specified source VLAN / NetworkInterface does not exist

**Parameters**

- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- srcNameId—InstanceNameId of port/EtherChannel or VLAN.
- modifiedDirection—modified SpanSourceDirection for the respective span source.

**Return Value**

void

modifySpanSources

Modifies list of SpanSource associations. This will enable users to modify the traffic direction for a list of existing VLAN/Interface to SPAN associations.

ValidationException is thrown if any of the following situation occurs:

- If spanSources is null or empty or it is not of type SpanSource.
- If any of the objects corresponding to SpanSource in spanSources does not exist
- If any of the objects in spanSources are not of LocalSpanSessionAppliesToNetworkInterface or LocalSpanSessionAppliesToVlan

**Parameters**

- opContext—Operational context
- spanSources—List of modified SpanSource associations

**Return Value**

void
removeDestinationPortsFromSpanSession

Removes destination ports/etherchannels from local SPAN session or remote SPAN destination session

ValidationException is thrown if any of the following situation occurs:

- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If NetworkInterface is removed completely if the session doesn't have another associations (filtered VLAN / source)
- If any of the objects corresponding to InstanceNameId in dstPortNameIds does not exist

**Parameters**

opContext—Operational context
dstPortNameIds—List of InstanceNameId of the interfaces to be removed from the given SPAN session.

**Return Value**

void

removeFilteredVlanRangeInSpanSession

Removes filtered VLANs from local SPAN session. Remote SPAN is not supported.

ValidationException is thrown if any of the following situation occurs:

- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If VLAN is removed completely if the session doesn't have another associations (source or destination)
- If any of the objects corresponding to InstanceNameId in dstPortNameIds does not exist

**Parameters**

opContext—Operational context
sessionNameId—InstanceNameId of a LocalSpanSession.
filteredVlans—range of VLAN IDs to be removed. Eg:"2,7-9".

**Return Value**

void

removeFilteredVlansInSpanSession

Removes filtered VLANs from local SPAN session. Remote SPAN is not supported.

ValidationException is thrown if any of the following situation occurs:

- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
removePortsFromSpanPermitList

Removes ports from SPAN destination permit list.
ValidationException is thrown if any of the following situation occurs:
- If neNameId is null or not InstanceNameId of the AbstractNetworkElement.
- If the specified AbstractNetworkElement does not exist
- If any of the element's InstanceNameId in interfaceNameIds is associated as SPAN destination to a SPAN session.

Validations:
14. Will not allow to remove the interface if is used as SPAN destination for any of the sessions in the corresponding device.

Parameters
opContext—Operational context
neNameId—Instance name ID of the network element.
interfaceNameIds—List of network interface instance name IDs

Return Value
void

removeSourcesFromSpanSession

Removes source ports/etherchannels or VLANs from a local SPAN session. Remote SPAN is not supported.
ValidationException is thrown if any of the following situation occurs:
- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If NetworkInterface or VLAN is removed completely if the session doesn't have any other associations (filtered VLAN / Destination port)
- If any of the objects corresponding to InstanceNameId in srcNameIds does not exist
removeSourcesInSpanSession

Remove source ports/etherchannels or VLANs from a local SPAN session. Remote SPAN session is not supported.

ValidationException is thrown if any of the following situation occurs:
- If sessionNameId is null or not InstanceNameId of the LocalSpanSession.
- If the specified LocalSpanSession does not exist
- If VLAN is removed completely if the session doesn't have any other associations (filtered VLAN / Destination port)

Parameters
- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- vlanRange—range of VLAN IDs to be configured as SPAN source. Eg: "2,7-9".

Return Value
- void

Parameters
- opContext—Operational context
- sessionNameId—InstanceNameId of LocalSpanSession.
- srcNameIds—List of InstanceNameId of VLAN / NetworkInterface.

Return Value
- void
StpApp Service

This chapter describes the DCNM web services’ API methods for the StpApp service.

Information About StpApp Service

This chapter defines the APIs exposed by the Spanning Tree Protocol (STP) service feature.

bindPvstSettingToVlans

Apply a PVST setting for one or more VLAN Ids. If any of the properties is set as null in PvstSetting, those properties will be set to default values.

ValidationException is thrown if any of the following situation occurs:

- If neIds is null
- If vlanIds is null.
- If setting is null.

Parameters

- opContext—Operational context
- neIds—NetworkElement instance name id collection.
- vlanIds—VLAN ids
- setting—Per-VLAN STP setting

Return Value

void

bindStpNetworkInterfaceSettingForMstInstances

Bind StpNetWorkInterfaceSettingBase to a range of MST ids for a given interface. The StpNetWorkInterfaceSettingBase instance needs to have a valid instance name set to it.

ValidationException is thrown if any of the following situation occurs:

- If networkInterfaceInstanceNameIds is null or it is empty.


**bindStpNetworkInterfaceSettingForVlans**

Bind `StpNetworkInterfaceSettingBase` to a range of VLAN ids for a given interface.

ValidationException is thrown if any of the following situation occurs:
- If `networkInterfaceInstanceNameIds` is null or it is empty.
- If `networkInterfaceInstanceNameIds` contains invalid network interface InstanceNameId or null value.
- If there is no equivalent `NetworkInterface` object with the given InstanceNameId in the `networkInterfaceInstanceNameIds`.
- If `mstIds` is null or it is empty.
- If `vlanIds` is null or it is empty.
- If the setting is null.

PropertiesException is thrown if any of the following situation occurs:
- If any of the `StpNetworkInterfaceSetting` attribute corresponding to an interface is not valid.

**Example:**
- cost of an `StpNetworkInterfaceSetting` should be some Integer with value between 1-200000000, any value that is less than 1 or greater than 200000000 will cause a `PropertiesException`.

**Parameters**
- `opContext`—Operational context
- `networkInterfaceInstanceNameIds`—List of InstanceNameId of network interfaces.
- `mstIds`—range of MST IDs (e.g. 1-10,20,35,50-120)
- `setting`—STP setting applied to network interfaces. Values range from 1-200000000.

**Return Value**
- void

---

**bindStpNetworkInterfaceSettingForVlans**

- If `networkInterfaceInstanceNameIds` contains invalid network interface InstanceNameId or null value.
- If there is no equivalent `NetworkInterface` object with the given InstanceNameId in the `networkInterfaceInstanceNameIds`.
- If `mstIds` is null or it is empty.
- If the setting is null.

PropertiesException is thrown if any of the following situation occurs:
- If any of the `StpNetworkInterfaceSetting` attribute corresponding to an interface is not valid.

**Example:**
- cost of an `StpNetworkInterfaceSetting` should be some Integer with value between 1-200000000, any value that is less than 1 or greater than 200000000 will cause a `PropertiesException`.

**Parameters**
- `opContext`—Operational context
- `networkInterfaceInstanceNameIds`—List of InstanceNameId of network interfaces.
send document comments to nexus7k-docfeedback@cisco.com

vlanIds—range of VLAN IDs (e.g. 1-10,20,35,50-120)
setting—STP setting applied to network interfaces. Values range from 1-200000000.

Return Value
void

bindStpNeworkInterfaceSettingsToNetworkInterface

Apply a STP setting to one or more network interfaces. The StpNetwrokInterfaceSetting should have a valid instance name set to it. Pass NULL values for the fields that need be reset to default values.

ValidationException is thrown if any of the following situation occurs:

- If networkInterfaceInstanceNameIds is null or it is empty.
- If networkInterfaceInstanceNameIds contains invalid network interface InstanceNameId or null value.
- If there is no equivalent NetworkInterface object with the given InstanceNameId in the networkInterfaceInstanceNameIds.
- If the setting is null.

PropertiesException is thrown if any of the following situation occurs:

- If any of the StpNetwrokInterfaceSetting attribute corresponding to an interface is not valid.

Example:

- cost of an StpNetwrokInterfaceSetting should be some Integer with value between 1-200000000, any value that is less than 1 or greater than 200000000 will cause a PropertiesException.

Parameters

- opContext—Operational context
- networkInterfaceInstanceNameIds—List of InstanceNameId of network interfaces.
- setting—STP setting applied to network interfaces. Values range from 1-200000000.

Return Value
void

bindVlansToMstInstance

Bind set of VLAN ids to MST instance.

ValidationException is thrown if any of the following situation occurs:

- If the mstId is null.
- If the mstId does not have equivalent MstSetting object in the database.
- If the vlanIds is null.

Parameters

- opContext—Operational context
createMstInstance

Creates an MST instance and associated VLANs in a network element.  

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If vlanInstanceNameIds is null or it is empty.
- If vlanInstanceNameIds contains invalid VLAN InstanceNameId or null value.
- If there is no equivalent VLAN object with the given InstanceNameId in the vlanInstanceNameIds.
- If the mstInstance is null.

PropertiesException is thrown if any of the following situation occurs:

- If any of the MstSetting attribute is not valid.

Example:

- stpHelloTime of an MstSetting should be some Integer with value between 1-10, any value that is less than 1 or greater than 10 will cause a PropertiesException.

IntegrityException is thrown if any of the following situation occurs:

- If the mstInstance already exists in the database.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of network element.
- mstInstance—MST instance to be created.
- vlanInstanceNameIds—List of InstanceNameId of VLANs to be associated to the MST instance.

Return Value

InstanceNameId of the new MST instance.

createMstInstanceByRange

Create a MST instance on a set of network elements. Creates a MST instance with the specified ID and associates it to the specified VLAN range. If the specified VLAN doesn’t exist, the VLAN will not be created but the mapping will be created.

Following associations are to be created for the given MST setting in the server:

- MST to VLAN mapping
ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIds is null or it is empty.
- If neInstanceNameIds contains invalid network element InstanceNameId or null value.
- If there is no equivalent network element object with the given InstanceNameId in the neInstanceNameIds.
- If the instanceId is null.
- If the vlanRange is null.

PropertiesException is thrown if any of the following situation occurs:

- If any of the MstSetting attribute is not valid.

Example:

instanceId of a MstSetting should be some Integer with value between 0-4094, any value that is less than 0 or greater than 4094 will cause a PropertiesException.

IntegrityException is thrown if any of the following situation occurs:

- If the MstSetting corresponding to instanceId already exist in the database.

**Parameters**

- opContext—Operational context
- neInstanceNameIds—List of InstanceNameId of network elements on which the MST instance is to be created.
- instanceId—MST instance ID.
- vlanRange—Range of VLAN IDs. If VLAN doesn't exist, then create a VLAN with instance state specified as incomplete.

**Return Value**

List of InstanceNameId for each MST instance created on the network elements.

---

createMstInstances

Create MST instances with their associations on one or more network elements.

Following associations are to be provided for the given MST setting:

- Network Element
- VLANs association that are mapped to an MST instance

ValidationException is thrown if any of the following situation occurs:

- If the mstInstances is null or the collection is empty.
- If the mstInstances contains one or more null element, or the collection contains objects that are not of type MstSetting.
- If the any of the MstSetting object within mstInstances have an association with AbstractNetworkElement which does not exist in database.

PropertiesException is thrown if any of the following situation occurs:

- If any of the MstSetting attribute is not valid.

Example:
createMstInstancesInNetworkElement

Creates MST instances in a network element. MST cannot be created without VLAN as so cation. Following associations are to be provided for the given MST setting:

- VLANs association that are mapped to an MST instance

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the mstInstances is null or the collection is empty.
- If the mstInstances contains one or more null element, or the collection contains objects that are not of type MstSetting.

PropertiesException is thrown if any of the following situation occurs:

- If any of the MstSetting attribute is not valid.

Example:

- stpHelloTime of an MstSetting should be some Integer with value between 1-10, any value that is less than 1 or greater than 10 will cause a PropertiesException.

IntegrityException is thrown if any of the following situation occurs:

- If the mstInstances contains a MstSetting that already exist in the database.

Parameters

- opContext—Operational context
- mstInstances—List of MST instances to be created.

Return Value

List of InstanceNameId of new MST instances.

createMstInstancesInNetworkElement
deleteMstInstances

Deletes a MST instance. If operational STP type is MST, all the VLANs corresponding to the MST instance will be assigned the default MST instance 0.

ValidationException is thrown if any of the following situation occurs:
- If mstInstanceNameIds collection is null or it is empty.
- If mstInstanceNameIds collection contains an element that is not of type MstSetting InstanceNameId.
- If mstInstanceNameIds collection contains a MstSetting that does not exist in the database.

Parameters

opContext—Operational context
mstInstances—List of InstanceNameId of MST instances to be deleted.

Return Value

void

getBlockingStpPortsCountInMsts

Returns list of Integer objects containing the number of ports with STP blocking state participating in each MST for the given list of MST instance ids.

ValidationException is thrown if any of the following situation occurs:
- If mstInstanceNameIds is null or it is empty.
- If mstInstanceNameIds contains invalid MstSetting InstanceNameId or null value.
- If there is no equivalent MstSetting object with the given InstanceNameId in the mstInstanceNameIds.

Parameters

opContext—Operational context
mstInstanceNameIds—List of InstanceNameId of MST ids

Return Value

List of Integers with count for the given list of MST Instance ids.

getBlockingStpPortsCountInVlans

Returns list of Integer objects containing the number of ports with STP blocking state participating in each VLAN for the given list of VLAN instance ids.

ValidationException is thrown if any of the following situation occurs:
- If vlanInstanceNameIds is null or it is empty.
- If vlanInstanceNameIds contains invalid VLAN InstanceNameId or null value.
- If there is no equivalent VLAN object with the given InstanceNameId in the vlanInstanceNameIds.
getForwardingStpPortsCountInMsts

Returns list of Integer objects containing the number of ports with STP forwarding state participating in each MST for the given list of MST instance ids.

ValidationException is thrown if any of the following situation occurs:
- If mstInstanceNameIds is null or it is empty.
- If mstInstanceNameIds contains invalid MstSetting InstanceNameId or null value.
- If there is no equivalent MstSetting object with the given InstanceNameId in the mstInstanceNameIds.

Parameters
opContext—Operational context
mstInstanceNameIds—List of InstanceNameId of MST ids

Return Value
List of Integers with count for the given list of MST Instance ids.

getForwardingStpPortsCountInVlans

Returns list of Integer objects containing the number of ports with STP forwarding state participating in each VLAN for the given list of VLAN instance ids.

ValidationException is thrown if any of the following situation occurs:
- If vlanInstanceNameIds is null or it is empty.
- If vlanInstanceNameIds contains invalid VLAN InstanceNameId or null value.
- If there is no equivalent VLAN object with the given InstanceNameId in the vlanInstanceNameIds.

Parameters
opContext—Operational context
vlanInstanceNameIds—List of InstanceNameId of VLANs

Return Value
List of Integers with count for the given list of VLAN ids.
getMstGlobalSettings

Returns MST global setting for given list of network elements. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIds contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIds.

**Parameters**

- opContext—Operational context
- neInstanceNameIds—List of InstanceNameId of network elements. If null, returns MST global setting for all network elements.

**Return Value**

List of MstGlobalSetting corresponding to network elements queried.

getMstIdsInNetwork

Returns IDs of MSTs configured on a list of network elements. No associations will be populated in this API, only MST range will be returned by this API. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIds contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIds.

**Parameters**

- opContext—Operational context
- neInstanceNameds—InstanceNameIds of network elements whose MST IDs are to be queried. If null, will return the MST ID from all the managed network elements.

**Return Value**

Range of MST IDs

getMstInstances

Returns MST instances corresponding to given InstanceNameIds. ValidationException is thrown if any of the following situation occurs:

- If mstSettingInstanceNameIds is null or it is empty.
- If mstSettingInstanceNameIds contains invalid MstSetting InstanceNameId or null value.
- If there is no equivalent MstSetting object with the given InstanceNameId in the mstSettingInstanceNameIds.
getMstInstancesInNetwork

Returns MST settings corresponding to a list of MST IDs for a list of network elements.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameIds contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIds.

**Parameters**

- opContext—Operational context
- mstIds—range of MST IDs (e.g. 1-10,20,35,50-120). If null, returns all MSTs.
- neInstanceNameIds—Instance name IDs of network elements for which MST IDs are to be queried. If null, returns MSTs on all managed network elements.

**Return Value**

List of MST settings

Following associations will be available for each MSTSetting:
- VLANs that are mapped
- Root bridgeId

Following associations will be available for each VLAN:
- VLAN status

getNetworkInterfacesWithStpSettings

Returns all the ports with STP settings.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameId.
getOperationalStpSettingForVlans

Returns a list of StpSetting (either PvstSetting or MstSetting) objects corresponding to the VLAN instance name id. ValidationException is thrown if any of the following situation occurs:

- If vlanInstanceNameIds contains invalid VLAN InstanceNameId or null value.
- If there is no equivalent VLAN object for the input VLAN InstanceNameId.

Parameters
- opContext—Operational context
- vlanInstanceNameIds—list of VLAN InstanceNameId.

Return Value
List of StpSetting object (MstSetting, PvstSetting) corresponding to the input VLAN object.

getStpDisabledVlansInNetwork

Returns STP Disabled VLAN objects corresponding to a list of VLAN IDs for a list of network elements. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIds contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIds.

Parameters
- opContext—Operational context
- vlanIds—range of VLAN IDs (e.g. 1-10,20,35,50-120). If null, returns all VLANs.
- neInstanceNameIds—Instance name IDs of network elements for which VLAN IDs are to be queried. If null, returns VLANs on all managed network elements.
getStpEnabledVlansInNetwork

Returns STP Enabled VLAN objects corresponding to a list of VLAN IDs for a list of network elements. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIds contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIds.

Parameters

- opContext—Operational context
- vlanIds—range of VLAN IDs (e.g. 1-10,20,35,50-120). If null, returns all VLANs.
- neInstanceNameIds—Instance name IDs of network elements for which VLAN IDs are to be queried. If null, returns VLANs on all managed network elements.

Return Value

List of ExternalVlan
Following associations will be available for each VLAN:

- STP Setting
- Secondary VLANs associated if the VLAN is a primary VLAN
- VLAN status

Following associations will be available for each PvstSetting:

- Root bridgeId

getStpGlobalSettings

Returns STP global setting for given list of network elements. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIds contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIds.

Send document comments to nexus7k-docfeedback@cisco.com
**getStpMode**

Returns list of STP type for a given list of network elements.

ValidationException is thrown if any of the following situation occurs:

- If `neInstanceNameIds` is null or it is empty.
- If `neInstanceNameIds` contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the `neInstanceNameIds`.

**Parameters**

- `opContext`—Operational context
- `neInstanceNameIds`—List of InstanceNameId of network elements.

**Return Value**

List of STP type of network elements.

**getStpNetworkInterfaceSettings**

Returns STP setting for give list of switch ports.

ValidationException is thrown if any of the following situation occurs:

- If `networkInterfaceInstanceNameIds` is null or it is empty.
- If `networkInterfaceInstanceNameIds` contains invalid NetworkInterface InstanceNameId or null value.
- If there is no equivalent NetworkInterface object with the given InstanceNameId in the `networkInterfaceInstanceNameIds`.

**Parameters**

- `opContext`—Operational context
- `networkInterfaceInstanceNameIds`—List of InstanceNameId of network interfaces.

**Return Value**

List of StpNetworkInterfaceSetting
getStpNetworkInterfaceSettingsForAccessPortsInMst

Returns list of SwitchedNetworkInterface objects corresponding to the input MST InstanceNameId. The STP setting information that exist between the input MST and any of its associated SwitchedNetworkInterfaces, will be available as a collection of StpNetworkInterfaceSettingForMst objects in each SwitchedNetworkInterface object. The StpNetworkInterfaceSettingForMst collection will be null when no STP setting information is available with the SwitchedNetworkInterface corresponding to the input MST.

ValidationException is thrown if the argument passed is null or it is not a valid MstSetting InstanceNameId.

Parameters
opContext—Operational context
mstInstanceNameId—InstanceNameId of MST.

Return Value
List of SwitchedNetworkInterface
Following associations will be available for each SwitchedNetworkInterface:
• Collection of StpNetworkInterfaceSettingForMst
• StpNetworkInterfaceStatus

Following associations will be available for each SwitchedNetworkInterface:
• Collection of StpNetworkInterfaceSettingForMst
• StpNetworkInterfaceStatus

getStpNetworkInterfaceSettingsForPortsInVlan

Returns list of SwitchedNetworkInterface objects corresponding to a VLAN. The STP setting information that exist between the input VLAN and any of the SwitchedNetworkInterfaces, will be available as a collection of StpNetworkInterfaceSettingForVlan objects in each SwitchedNetworkInterface object. The StpNetworkInterfaceSettingForVlan collection will be null when no STP setting information is available with the SwitchedNetworkInterface corresponding to the input VLAN.

ValidationException is thrown if the argument passed is null or it is not a valid VLAN InstanceNameId.

Parameters
opContext—Operational context
vlanInstanceNameId—InstanceNameId of VLAN.

Return Value
List of SwitchedNetworkInterface
Following associations will be available for each SwitchedNetworkInterface:
• StpNetworkInterfaceSettingForVlan
• StpNetworkInterfaceStatus
getStpNetworkInterfaceSettingsForPortsInVlansByInterfaces

Returns list of SwitchedNetworkInterface objects which has an StpNetworkInterfaceSettingVlan association existing between each VLAN in vlanRange and any of the SwitchedNetworkInterface object corresponding to ifInstanceNameIdCol. The STP setting information that exist between the input VLAN and SwitchedNetworkInterfaces corresponding to ifInstanceNameIdCol, will be available as a collection of StpNetworkInterfaceSettingForVlan objects in each SwitchedNetworkInterface object. The StpNetworkInterfaceSettingForVlan collection will be null when no STP setting information is available with the SwitchedNetworkInterface corresponding to the input VLAN.

ValidationException is thrown if the argument passed is null or the vlanRange size is lesser than or equal to 0.

Parameters
- opContext—Operational context
- ifInstanceNameIdCol—of SwitchedNetworkInterface
- vlanRange—Range of VLANs.

Return Value
List of SwitchedNetworkInterface

Following associations will be available for each SwitchedNetworkInterface:
- StpNetworkInterfaceSettingForVlan
- StpNetworkInterfaceStatus

getStpNetworkInterfaceSettingsForTrunkPortsInMst

Returns list of SwitchedNetworkInterface objects corresponding to the input MST InstanceNameId. The STP setting information that exist between the input MST and any of its associated SwitchedNetworkInterfaces, will be available as a collection of StpNetworkInterfaceSettingForMst objects in each SwitchedNetworkInterface object. The StpNetworkInterfaceSettingForMst collection will be null when no STP setting information is available with the SwitchedNetworkInterface corresponding to the input MST.

ValidationException is thrown if the argument passed is null or it is not a valid MstSetting InstanceNameId.

Parameters
- opContext—Operational context
- mstInstanceNameId—InstanceNameId of MST.

Return Value
List of SwitchedNetworkInterface

Following associations will be available for each SwitchedNetworkInterface:
- Collection of StpNetworkInterfaceSettingForMst
- StpNetworkInterfaceStatus
getStpNetworkInterfaceSettingsForVlanMemberPortsInMst

Returns list of SwitchedNetworkInterface objects corresponding to the input MST InstanceNameId and member ports of each VLAN corresponding to vlanRange. The STP setting information that exist between the input MST and any of the VLAN member ports will be available as a collection of StpNetorkInterfaceSettingForMst objects in each SwitchedNetworkInterface object. The StpNetorkInterfaceSettingForMst collection will be null when no STP setting information is available with the SwitchedNetworkInterface corresponding to the input MST.

ValidationException is thrown if the argument passed is null or it is not a valid MstSetting InstanceNameId.

Parameters

- opContext—Operational context
- mstInstanceNameId—InstanceNameId of MST
- vlanInstanceNameId—InstanceNameId of VLAN

Return Value

List of SwitchedNetworkInterface

Following associations will be available for each SwitchedNetworkInterface:

- Collection of StpNetorkInterfaceSettingForMst
- StpNetorkInterfaceStatus

getStpSettingsForMst

Returns STP setting for a MST on a list of network elements. For a given MST and a list of network elements, returns a List of MstSetting, returns STP setting for all network elements.

ValidationException is thrown if any of the following situation occurs:

- If mstId is null.
- If neInstanceNameIds contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIds.

Parameters

- opContext—Operational context
- mstId—ID of MST instance queried.
- neInstanceNameIds—List of InstanceNameId of network elements queried.

Return Value

List of MstSetting with associated StpNetRootBridgeStatus. Root bridge status will be null if no port in MST is operational

Following associations will be available for each MstSetting:

- VLANs that are mapped
- Root bridgeId
Following associations will be available or each VLAN:

- VLAN status

### getStpSettingsForVlan

Returns STP setting (PVRST) for a VLAN on a set of network elements. For a given VLAN and a list of network elements, returns a List of PvstSetting. If network elements are not specified, returns STP setting for all network elements.

ValidationException is thrown if any of the following situation occurs:

- If vlanId is null.
- If neInstanceNameIds contains invalid NetworkElement InstanceNameId or null value.
- If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIds.

**Parameters**

- `opContext`—Operational context
- `vlanId`—ID of VLAN queried.
- `neInstanceNameIds`—List of InstanceNameId of network elements queried.

**Return Value**

A List of PvstSetting with associated StpRootBridgeStatus. Root bridge status will be null if no port in VLAN is operational.

Following associations will be available for each PvstSetting:

- Root bridgeId

### getStpStatusForNetworkInterfacesInVlan

Returns a list of StpNetWorkInterfaceStatus objects corresponding to the StpNetWorkInterfaceSettingForVlan association that exists between the VLAN corresponding to `vlanInstanceNameId` and each instance of NetworkInterface corresponding to `interfaceInstanceNameIds` list. An empty list will be returned if the device is operating in mst mode. ValidationException is thrown if any of the following situation occurs:

- If `vlanInstanceNameId` contains invalid VLAN InstanceNameId or null value.
- If there is no equivalent VLAN object for the input VLAN InstanceNameId.
- If `interfaceInstanceNameIds` contains invalid NetworkInterface InstanceNameId or null value.
- If there is no equivalent NetworkInterface object for the input NetworkInterface InstanceNameId.

**Parameters**

- `opContext`—Operational context
- `vlanInstanceNameId`—VLAN InstanceNameId.
- `interfaceInstanceNameIds`—list of NetworkInterface InstanceNameId.
Chapter 33  StpApp Service

Send document comments to nexus7k-docfeedback@cisco.com

return value
List of StpNetworKInterfaceStatus.

getTotalStpPortsCountInMsts

Returns list of Integer objects containing the number of ports with STP status participating in each MST for the given list of MST instance ids.

ValidationError is thrown if any of the following situation occurs:

- If mstInstanceNameIds is null or it is empty.
- If mstInstanceNameIds contains invalid MstSetting InstanceNameId or null value.
- If there is no equivalent MstSetting object with the given InstanceNameId in the mstInstanceNameIds.

Parameters

opContext—Operational context
mstInstanceNameIds—List of InstanceNameId of MST ids

Return Value
List of Integers with count for the given list of MST Instance ids.

getTotalStpPortsCountInVlans

Returns list of Integer objects containing the number of ports with STP status participating in each VLAN for the given list of VLAN instance ids.

ValidationError is thrown if any of the following situation occurs:

- If vlanInstanceNameIds is null or it is empty.
- If vlanInstanceNameIds contains invalid VLAN InstanceNameId or null value.
- If there is no equivalent VLAN object with the given InstanceNameId in the vlanInstanceNameIds.

Parameters

opContext—Operational context
vlanInstanceNameIds—List of InstanceNameId of VLANs.

Return Value
List of Integers with count for the given list of VLAN ids.

getVlansWithPvrstSetting

Returns VLANs with PVRST Settings corresponding to a list of VLAN IDs for a list of network elements.

ValidationError is thrown if any of the following situation occurs:

- If neInstanceNameIds contains invalid NetworkElement InstanceNameId or null value.
If there is no equivalent NetworkElement object with the given InstanceNameId in the neInstanceNameIds.

**Parameters**

- **opContext**—Operational context
- **vlanIds**—range of VLAN IDs (e.g. 1-10,20,35,50-120). If null, returns all VLANs.
- **neInstanceNameIds**—Instance name IDs of network elements for which VLAN IDs are to be queried. If null, returns MSTs on all managed network elements.

**Return Value**

A List of VLAN

Following associations will be available for each VLAN:

- STP Setting
- Secondary VLANs associated if the VLAN is a primary VLAN.
- Primary VLAN reference incase of Secondary VLAN.
- VLAN status

Following associations will be available for each PvstSetting:

- Root bridgeId

Following associations will be available for each SecondaryVlan:

- STP Setting. This settings has StpRootBridgeStatus
- VLAN status

### modifyMstGlobalSettings

Modify MST global setting for one or more network elements.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the modifiedMstGlobalSetting is null.

**Parameters**

- **opContext**—Operational context
- **modifiedMstGlobalSetting**—Modified MstGlobalSetting.

**Return Value**

void

### modifyMstInstances

Modify settings for MST instances. User cannot modify the MST id in the settings. User also cannot change both switch priority and STP root type. Can set either switch priority or STP root type.
modifyPvstSettings

Modify settings for PVRST instances. User also cannot change both switch priority and STP root type. Can set either switch priority or STP root type.

ValidationException is thrown if any of the following situation occurs:

- If the pvstSettings is null or the collection is empty.
- If the pvstSettings contains one or more null element, or the collection contains objects that are not of type PvstSetting.
- If the any of the PvstSetting object within pvstSettings does not have equivalent persisted PvstSetting object.

PropertiesException is thrown if any of the following situation occurs:

- If any of the PvstSetting attribute is not valid.

Example:

- stpHelloTime of an PvstSetting should be some Integer with value between 1-10, any value that is less than 1 or greater than 10 will cause a PropertiesException.

Parameters

- opContext—Operational context
- pvstSettings—List of modified PvstSetting.

Return Value

- void
modifyStpGlobalSettings

Modify STP global setting for a network element. User can modify the protocol also using the API. ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null.
- If neInstanceNameId is not a valid network element InstanceNameId.
- If the modifiedStpGlobalSetting is null.

Parameters
opContext—Operational context
modifiedStpGlobalSetting—Modified StpGlobalSetting.

Return Value
void

modifyStpNetworkInterfaceSettingForMst

Modify MST settings for a specific MST Instance for a switch port. Set NULL values for the fields that need be reset to default values.

ValidationException is thrown if any of the following situation occurs:

- If settings is null or it is empty.

PropertiesException is thrown if any of the following situation occurs:

- If any of the StpNetNetworkInterfaceSettingForMst attribute corresponding to an interface is not valid.

Example:

- cost of an StpNetNetworkInterfaceSettingForMst should be some Integer with value between 1-200000000, any value that is less than 1 or greater than 200000000 will cause a PropertiesException.

Parameters
opContext—Operational context
settings—List of StpNetNetworkInterfaceSettingForMst

Return Value
void

modifyStpNetworkInterfaceSettingForVlan

Modify PVLAN settings for a specific VLAN for a switch port. Set NULL values for the fields that need be reset to default values.

ValidationException is thrown if any of the following situation occurs:

- If settings is null or it is empty.

PropertiesException is thrown if any of the following situation occurs:
modifyStpNetworkInterfaceSettings

Modify STP settings of network interfaces. Pass NULL values for the parameters that need be reset to default values.

ValidationException is thrown if any of the following situation occurs:

- If stpNetworkInterfaceSettings is null or it is empty.
- If stpNetworkInterfaceSettings contains invalid StpNetworkInterfaceSetting InstanceNameId or null value.

PropertiesException is thrown if any of the following situation occurs:

- If any of the StpNetworkInterfaceSetting attribute is not valid.

Example:

- cost of an StpNetworkInterfaceSetting should be some Integer with value between 1-200000000, any value that is less than 1 or greater than 200000000 will cause a PropertiesException.

Parameters

opContext—Operational context

stpNetworkInterfaceSettings—List of modified STP settings.

Return Value

void

modifyStpSettingForNetworkInterfaces

Modify link setting and STP setting for a given collection of Network Interfaces. Pass NULL values for the fields that need be reset to default values.

ValidationException is thrown if any of the following situation occurs:

- If networkInterfaces is null or it is empty.
- If the any of the NetworkInterface object within networkInterfaces does not have equivalent persisted NetworkInterface object.

Parameters

networkInterfaces—List of Network Interfaces.

Return Value

void
modifyStpType

Modify STP type for a set of network elements.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIds is null or it is empty.
- If neInstanceNameIds contains invalid network element InstanceNameId or null value.
- If there is no equivalent network element object with the given InstanceNameId in the neInstanceNameIds.
- If the protocol is null.

Parameters

opContext—Operational context

neInstanceNameIds—List of InstanceNameId of network elements.

protocol—STP protocol type.

Return Value

void

unbindVlansFromMstInstance

Unbind one or more VLANs from MST instance.

ValidationException is thrown if any of the following situation occurs:

- If the mstId is null.
- If the mstId does not have equivalent MstSetting object in the database.
- If the vlanIds is null.

IntegrityException is thrown if any of the following situation occurs:

PropertiesException is thrown if any of the following situation occurs:

- If any of the StpNetworkInterfaceSetting or StpLinkSetting attribute corresponding to an interface is not valid.

Example:

- cost of an StpNetworkInterfaceSetting should be some Integer with value between 1-200000000, any value that is less than 1 or greater than 200000000 will cause a PropertiesException.

Parameters

opContext—Operational context

networkInterfaces—List of NetworkInterface with modified StpLinkSetting and StpNetNetworkInterfaceSetting

Return Value

void
unbindVlansFromMstInstance

- If MstSetting object does not have any VLAN associated with it.
- If all VLANs associated to MstSetting object is specified in unbind vlanIds.

FeatureException is thrown if MstSetting object corresponds to default MST.
ParameterException is thrown if any of the VLANs specified in vlanIds has no pre-existing association with MstSetting object.

Parameters

- opContext—Operational context
- mstId—InstanceNameId of MST
- vlanIds—range of VLAN IDs (e.g. 1-10,20,35,50-120)

Return Value

void
TrafficStormControlApp Service

This chapter describes the DCNM web services’ API methods for the TrafficStormControlApp service.

Information About TrafficStormControlApp Service

This chapter defines the APIs exposed by the traffic storm control service feature.

disableBroadcastStormControlInInterfaces

Disables broadcast storm control in a collection of interfaces, given the InstanceNameId of the interfaces. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface. Enabling/Disabling traffic storm control is not possible in those interfaces.

ValidationException is thrown if any of the following situation occurs:

• If the argument passed interfaceInstanceNameIds is null or it is not a valid interface InstanceNameId.
• If the interface does not exist in the database.

PropertiesException is thrown if the BroadcastSuppression capability of the given interface is None.

Parameters

opContext— Operational context.
interfaceInstanceNameIds— A collection of InstanceNameId of interfaces in which broadcast storm control has to be disabled.

Return Value

void
disableMulticastStormControlInInterfaces

Disables multicast storm control in a collection of interfaces, given the InstanceNameId of the interfaces. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface. Enabling/Disabling traffic storm control is not possible in those interfaces.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed interfaceInstanceNameIds is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.

PropertiesException is thrown if the BroadcastSuppression capability of the given interface is None.

Parameters

- opContext--- Operational context.
- interfaceInstanceNameIds--- A collection of InstanceNameId of interfaces in which multicast storm control has to be disabled.

Return Value

- void

disableStormControlInInterfaces

Disables storm control for all three types of traffics (unicast, broadcast, or multicast) in a given collection of interfaces. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface. Enabling/Disabling traffic storm control is not possible in those interfaces.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed interfaceInstanceNameIds is null or empty
- If any of the element in interfaceInstanceNameIds is not a valid interfaceInstanceNameId.
- If the interface does not exist in the database.

Parameters

- opContext--- Operational context.
- interfaceInstanceNameId--- InstanceNameId of the interfaces in which storm control for all three traffic types (unicast, broadcast and multicast) is to be disabled.

Return Value

- void
disableUnicastStormControlInInterfaces

Disables unicast storm control in a collection of interfaces, given the InstanceNameId of the interfaces. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface. Enabling/Disabling traffic storm control is not possible in those interfaces.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed interfaceInstanceNameIds is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.
- PropertiesException is thrown if the BroadcastSuppression capability of the given interface is None.

**Parameters**

- opContext— Operational context.
- interfaceInstanceNameIds—- A collection of InstanceNameId of interfaces in which unicast storm control has to be disabled.

**Return Value**

void

enableBroadcastStormControlInInterfaces

Enables broadcast storm control in a collection of interfaces with the given threshold. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface. Enabling/Disabling traffic storm control is not possible in those interfaces.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed interfaceInstanceNameIds is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.
- If the threshold value is 100, and traffic storm control for other types (unicast or multicast) were also not enabled already.
- PropertiesException is thrown if the BroadcastSuppression capability of the given interface is None.

**Parameters**

- opContext— Operational context.
- interfaceInstanceNameIds—- A collection of InstanceNameId of interfaces in which broadcast is to be enabled.
- threshold—- A Float object indicating the value of storm control threshold. This object shall have a float value less than 100 and greater than 0. This is an optional argument. If this argument is set to null, already configured threshold in the interfaces will be retained.

**Return Value**

void
enableMulticastStormControlInInterfaces

Enables multicast storm control in a collection of interfaces with the given threshold. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface. Enabling/Disabling traffic storm control is not possible in those interfaces.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed interfaceInstanceId is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.
- If the threshold value is 100, and traffic storm control for other types (unicast or broadcast) were also not enabled already.

PropertiesException is thrown if the BroadcastSuppression capability of the given interface is None.

Parameters

- opContext—— Operational context.
- interfaceInstanceId—— A collection of InstanceNameId of interfaces in which * multicast is to enabled.
- threshold—— A Float object indicating the value of storm control threshold. This object shall have a float value less than 100 and greater than 0. This is an optional argument. If this argument is set to null, already configured threshold in the interfaces will be retained.

Return Value

void

enableUnicastStormControlInInterfaces

Enables unicast storm control in a collection of interfaces with the given threshold. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface. Enabling/Disabling traffic storm control is not possible in those interfaces.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed interfaceInstanceId is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.
- If the threshold value is 100, and traffic storm control for other types (broadcast or multicast) were also not enabled already.

PropertiesException is thrown if the BroadcastSuppression capability of the given interface is None.

Parameters

- opContext—— Operational context.
interfaceInstanceNameIds— A collection of InstanceNameId of interfaces in which unicast is to enabled.

threshold—- A Float object indicating the value of storm control threshold. This object shall have a float value less than 100 and greater than 0. This is an optional argument. If this argument is set to null, already configured threshold in the interfaces will be retained.

**Return Value**

void

---

**getInterfacesWithTrafficStormControlSettingInNetwork Element**

Returns all the traffic storm control enabled and disabled interfaces in the given network element. An interface is considered to be traffic storm control enabled, if storm control for one or more traffic types (unicast, broadcast or multicast) is enabled in it. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed neInstanceNameId is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

**Parameters**

- opContext—- Operational context.
- neInstanceNameId—- InstanceNameId of the network element.

**Return Value**

A collection of traffic storm control enabled and traffic storm control disabled NetworkInterface. The interfaces can be of type SwitchedNetworkInterface or RoutedNetworkInterface. In the returned list only the following associations will be present and other associations will be cleared.

- Traffic storm control association.
- Network interface name association.

---

**getTrafficStormControlDisabledInterfacesInNetworkElement**

Returns all the interfaces in which traffic storm control is disabled in the given network element. An interface is considered to be traffic storm control disabled, if none of the three traffic types (unicast, broadcast or multicast) is enabled and the threshold is 100. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface. The interfaces whose BroadcastSupressionCapability is None, will not be included in the returned list of this API.

ValidationException is thrown if any of the following situation occurs:
getTrafficStormControlSettingInInterfaces

Returns the traffic storm control setting in the collection of interfaces. For interfaces in which broadcast suppression capability has been set to None, NULL value will be returned by this API.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed is null or it is not a valid interface InstanceNameId.
- If the interface does not exist in the database.

Parameters

opContext—- Operational context.
interfaceInstanceIds—- A collection of InstanceNameId of interfaces.

Return Value

A collection of TrafficStormControlSetting objects pertaining to the interfaces.

getTrafficStormControlledInterfacesInNetworkElement

Returns all the traffic storm control enabled interfaces in the given network element. An interface is considered to be traffic storm control enabled, if storm control for one or more traffic types (unicast, broadcast or multicast) is enabled in it. If BroadcastSuppressionCapability is specified as None in the PortCapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface.

ValidationException is thrown if any of the following situation occurs:

- If the argument passed neInstanceNameId is null or it is not a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters

opContext—- Operational context.
neInstanceNameId—- InstanceNameId of the network element.
modifyStormControlInInterfaces

Modifies the traffic storm control setting in a given collection of interfaces. This API can also be used to enable or disable specify traffic control. When disabling all three (unicast, broadcast, multicast) traffic storm control, threshold should be 100. In the TrafficStormControlSetting objects passed, user shall not specify 100% threshold and specify enable state for one of the traffic types. If BroadcastSuppressionCapability is specified as None in the Portcapability object of an interface, those interfaces will not have traffic storm control configurations, though they are of type SwitchedNetworkInterface or RoutedNetworkInterface.

ValidationException is thrown if any of the following situation occurs:
- If the argument passed trafficStormControlSettings is null or empty.
- If any of the element in the collection trafficStromControlSettings is not of type TrafficStormControlSetting.
- If any of the element in the collection trafficStormControlSettings does not exist in the database.
- If the attributes of any of the element in the collection trafficStromControlSettings is not valid.

Example:
- Bandwidth threshold is 100, when storm control for any one of the three traffic types (unicast, broadcast or multicast) is enabled.
- Bandwidth threshold is not 100, when storm control for all three traffic types is disabled.

PropertiesException is thrown if the bandwidth threshold is not in the range 0—100.

Parameters
opContext—— Operational context.
trafficStormControlSettings—— Collection of modified traffic storm control setting objects.

Return Value
void
TunnelApp Service

This chapter describes the DCNM web services’ API methods for the TunnelApp service.

Information About TunnelApp Service

Tunneling allows you to encapsulate arbitrary packets inside a transport protocol. This feature is implemented as a virtual interface to provide a simple interface for configuration. The tunnel interface is designed to provide the services necessary to implement any standard point-to-point encapsulation scheme. Because tunnels are point-to-point links, a separate tunnel needs to be configured for each link. The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Modify APIs—Modify existing IpGreTunnelNetworkInterface settings.
- Delete APIs—Delete existing tunnel interfaces in the device.
- Apply APIs—Apply a setting to a group of tunnel interface settings.

bindKeepAliveSettingToNetworkInterfaces

Modifies KeepAliveSetting to one or more Tunnel Interfaces.

ValidationException is thrown if any of the following situation occurs:

- If tunnelInstanceIds is null or empty or it is not of type InstanceNameId.
- If tunnelInstanceIds is not a valid InstanceNameId of TunnelNetworkInterface object.
- If keepAliveSetting is null.

Parameters

- opContext—Operational context
- tunnelInstanceIds—List of one or more InstanceNameId of TunnelNetworkInterface.
- keepAliveSetting—KeepAliveSetting to be applied to TunnelNetworkInterface objects.

Return Value

- void
bindPathMtuDiscoverySettingToNetworkInterfaces

Modifies PathDiscoverySetting to one or more Tunnel Interfaces

ValidationException is thrown if any of the following situation occurs:

- If tunnelInstanceIds is null or empty or it is not of type InstanceNameId.
- If tunnelInstanceIds is not a valid InstanceNameId of TunnelNetworkInterface object.
- If keepAliveSetting is null.

Parameters

- opContext—Operational context
- tunnelInstanceIds—List of one or more InstanceNameId of TunnelNetworkInterface
- pathMtuDiscoverySetting—PathMtuDiscoverySetting to be applied to TunnelNetworkInterface objects

Return Value

- void

createIpGreTunnelNetworkInterfaces

Create a list of tunnel interfaces in a network elements.

Following associations are to be provided for the given IpGreTunnelNetworkInterface:

- AbstractNetworkElementReference corresponding to the specified IpGreTunnelNetworkInterface
- AbstractNetworkElement reference

ValidationException is thrown if any of the following situation occurs:

- If tunnelIfs is null or empty or it is not a valid TunnelNetworkInterface object.
- If an instance of IpGreTunnelNetworkInterface exists in the database.

Parameters

- opContext—Operational context
- tunnelIfs—List of TunnelNetworkInterface objects to be created

Return Value

- List of InstanceNameIds corresponding to the new IpGreTunnelNetworkInterface

deleteIpGreTunnelNetworkInterfaces

Deletes one or more Tunnel InterfaceObjects. Given a InstanceNameId of TunnelNetworkInterface, the corresponding tunnel interface object is deleted from the server.

ValidationException is thrown if any of the following situation occurs:

- If tunnelInstanceIds collection is null or it is empty.
- If tunnelInstanceIds collection contains an element that is not of type TunnelNetworkInterface InstanceNameId.
disableTunnelService

Disables Tunnel service in the device. Given the instance name ID of one or more network elements, tunnel service will be disabled on those elements.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If neInstanceNameIdCol does not a valid AbstractNetworkElement InstanceNameId.

Parameters

opContext—Operational context
eoInstanceNameIdCol—List of InstanceNameId of tunnel service enabled AbstractNetworkElement

Return Value

void

enableTunnelService

Enables Tunnel service globally in the device. Given the instance name ID of one or more network elements, tunnel will be enabled on those elements.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If neInstanceNameIdCol does not a valid AbstractNetworkElement InstanceNameId.

Parameters

opContext—Operational context
neInstanceNameIdCol—List of InstanceNameId of tunnel service disabled AbstractNetworkElement

Return Value

void
getAllTunnelInterfacesInNetworkElement

Returns a list of TunnelNetworkInterface configured in the given device. Given the InstanceNameId of a network element, returns a collection of TunnelNetworkInterface in the network element.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null or it is not a valid InstanceNameId of a network element.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of the network element

Return Value

List of instances of TunnelNetworkInterface

getTunnelDestinationInterface

Returns instance of Destination NetworkInterface configured for the given tunnel.

ValidationException is thrown if any of the following situation occurs:

- If tunnelIf is null or it is not a valid InstanceNameId of an IpGreTunnelNetworkInterface.

Parameters

- opContext—Operational context
- tunnelIf—InstanceNameId of the IpGreTunnelNetworkInterface

Return Value

instances of NetworkInterface

getTunnelGlobalSetting

Returns a list of TunnelGlobalSetting corresponding to the given AbstractNetworkElement.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If neInstanceNameIdCol does not a valid AbstractNetworkElement InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—List of one or more InstanceNameId of AbstractNetworkElement

Return Value

List of TunnelGlobalSetting
getTunnelNetworkInterfaces

Returns the Tunnel Network Interface corresponding to the InstanceNameId. Given the InstanceNameId of one TunnelNetworkInterface, returns the corresponding tunnel Network Interface object.

ValidationException is thrown if any of the following situation occurs:

- If tunnelIfInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If tunnelIfInstanceNameIdCol is not a valid InstanceNameId of TunnelNetworkInterface object.

**Parameters**

- opContext—Operational context
- tunnelIfInstanceNameIdCol—List of one or more InstanceNameId of TunnelNetworkInterface objects.

**Return Value**

List of instances of TunnelNetworkInterface.

getTunnelSourceNetworkInterface

Returns instance of source NetworkInterface configured for the given tunnel.

ValidationException is thrown if any of the following situation occurs:

- If tunnelIf is null or it is not a valid InstanceNameId of a IpGreTunnelNetworkInterface.

**Parameters**

- opContext—Operational context
- tunnelIf—InstanceNameId of the IpGreTunnelNetworkInterface

**Return Value**

instances of NetworkInterface

modifyIpGreTunnelNetworkInterfaces

Modify a IpGreTunnelNetworkInterface Objects in one or more network elements.

ValidationException is thrown if any of the following situation occurs:

- If modifiedTunnelIfs is null or empty or it is not of type TunnelNetworkInterface.
- Only existing IpGreTunnelNetworkInterface (exist in database) can be modified.

PropertiesException is thrown if any of the following situation occurs:

- In the modifiedTunnelIfs collection, if any one of the TunnelNetworkInterface attribute is not valid.

Example:

- MTU of a Tunnel Interface is read-only. It Contains default value and it should not be editable.

**Parameters**

- opContext—Operational context
modifiedTunnelIifs—List of modified TunnelNetworkInterface objects

Return Value

void
UserApp Service

This chapter describes the DCNM web services’ API methods for the UserApp service.

Information About UserApp Service

This chapter defines the APIs exposed by the user service feature.

**bindUserToAllNetworkElements**

Binds the user to all network elements.

**Parameters**
- opContext—Operational context
- InstanceNameId—Application user

**Return Value**
- void

**bindUserToAllNetworkGroups**

Binds the user to all network groups.

**Parameters**
- opContext—Operational context
- InstanceNameId—Application user

**Return Value**
- void

**bindUserToNetworkElements**

Binds the user to the specified network elements.
bindUserToNetworkGroups

Binds the user to the specified network groups.

**Parameters**
- opContext—Operational context
- InstanceNameId—Application user
- InstanceNameId—List of Instance name IDs of network groups

**Return Value**
- void

createUsers

Creates users based on a list of application users.

**Parameters**
- opContext—Operational context
- ApplicationUser—List of application users

**Return Value**
- List of application users created.

deleteUsers

Deletes users based on a list of application users.

**Parameters**
- opContext—Operational context
- ApplicationUser—List of application users

**Return Value**
- void
getAllUsers

Gets the list of user information for all application users.

Parameters
opContext—Operational context

Return Value
List of ApplicationUser

getDefaultNetworkElementUserCredentials

Get the default network element user credentials.

Parameters
opContext—Operational context

Return Value
NetworkElementUserCredentials

getNetworkElementGroupsForUser

Gets the list of network element groups for the specified user.

Parameters
opContext—Operational context
InstanceNameId—Instance name ID of user

Return Value
List of NetworkElementGroup

getNetworkElementsForUser

Gets the list of network elements for the specified user.

Parameters
opContext—Operational context
InstanceNameId—Instance name ID of user

Return Value
List of AbstractNetworkElement
getPreferences

Returns the preferences for the user invoking the API.

**Parameters**

- opContext—Operational context

**Return Value**

Preferences

getUserCredentialsForNetworkElementGroups

Gets the list of user credentials for the specified network element groups.

**Parameters**

- opContext—Operational context
- InstanceNameId—Application user
- InstanceNameId—List of network element groups

**Return Value**

List of NetworkElementUserCredentials

getUserCredentialsForNetworkElements

Gets the list of user credentials for the specified network elements.

**Parameters**

- opContext—Operational context
- InstanceNameId—Application user
- InstanceNameId—List of network elements

**Return Value**

List of NetworkElementUserCredentials

getUsers

Gets the list of user information for the specified application users.

**Parameters**

- opContext—Operational context
- InstanceNameId—List of application users
modifyDefaultNetworkElementUserCredentials

Modifies the default user credentials to the value supplied.

**Parameters**
- opContext—Operational context
- NetworkElementUserCredentials—User credentials

**Return Value**
NetworkElementUserCredentials

modifyPreferences

Modifies the preferences for the user invoking the API.

**Parameters**
- opContext—Operational context
- Preferences—User preferences

**Return Value**
Preferences

modifyUserCredentialsForNetworkElementGroups

Modifies the credentials for the specified user and the network element groups.

**Parameters**
- opContext—Operational context
- InstanceNameId—User to modify
- InstanceNameId—List of network element groups to modify
- NetworkElementUserCredentials—User credentials

**Return Value**
void

modifyUserCredentialsForNetworkElements

Modifies the credentials for the specified user and the network elements.
modifyUsers

Modifies the list of specified application users.

**Parameters**
- opContext—Operational context
- ApplicationUser—List of users

**Return Value**
void

unbindUserFromAllNetworkElements

Unbinds the specified user from all network elements.

**Parameters**
- opContext—Operational context
- InstanceNameId—User to unbind

**Return Value**
void

unbindUserFromAllNetworkGroups

Unbinds the specified user from all network groups.

**Parameters**
- opContext—Operational context
- InstanceNameId—User to unbind

**Return Value**
void
unbindUserFromNetworkElements

Unbinds the specified user from all network elements.

Parameters

- **opContext**—Operational context
- **InstanceNameId**—User to unbind
- **InstanceNameId**—List of network elements

Return Value

void

unbindUserFromNetworkGroups

Unbinds the specified user from all network groups.

Parameters

- **opContext**—Operational context
- **InstanceNameId**—User to unbind
- **InstanceNameId**—List of network groups

Return Value

void
unbindUserFromNetworkGroups

Send document comments to nexus7k-docfeedback@cisco.com
VdcApp Service

This chapter describes the DCNM web services’ API methods for the VdcApp service.

Information About VdcApp Service

A physical device can be configured to look like multiple virtual devices called virtual device contexts (VDCs). A Layer 2, Layer 3, or port channel device interface can be assigned to one and only one VDC. VDCs interact with each other by running protocols on interfaces on which they are interconnected. Each VDC performs all the functions of the physical device. The API categories are as follows:

- Query/Get APIs—Query data from the persistent database.
- Modify APIs—Modify existing VDCs and resource templates.
- Delete APIs—Delete VDCs or resource templates in the device.

bindPortsToVdcs

Binds a list of Ports to the specified VirtualNetworkElement

ValidationException is thrown if any of the following situation occurs:

- If networkInterfaceInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If vdcInstanceNameId is null or empty or it is not of type InstanceNameId.
- Only existing VirtualNetworkElement/NetworkInterface (exist in database) can be modified.

Parameters

- opContext—Operational context
- vdcInstanceNameId—InstanceNameId of VDC to which the corresponding NetworkInterface objects has to be configured.
- networkInterfaceInstanceNameIdCol—List of InstanceNameId of NetworkInterface which will be configured to the given VDC.

Return Value

List of modified NetworkInterface objects that are persisted in the database.
bindVdcsWithResourceTemplate

Creates VdcResourceLimit from the specified ResourceTemplate and binds with the given VDCs.

ValidationException is thrown if any of the following situation occurs:

- If vdcInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If resourceTemplateId is null or empty or it is not of type InstanceNameId.
- Only existing VirtualNetworkElement (exist in database) can be modified.

Parameters

- opContext—Operational context
- resourceTemplateId—InstanceNameId of resourceTemplate to be configured along with VDC.
- vdcInstanceNameIdCol—List of InstanceNameId of VDCs which will be applied with the given resource template.

Return Value

List of that are persisted with the specified resource limits. The returned collection will contain only the following associations and the rest will be cleared:

- VdcResourceLimit corresponding to VirtualNetworkElement
- VirtualNetworkElementStatus corresponding to VirtualNetworkElement
- NetworkInterface present in the VirtualNetworkElement
- AbstractNetworkElementStatus present in the VirtualNetworkElement
- SshServerSetting corresponding to the VirtualNetworkElement

createResourceTemplates

Creates resource templates in the specified device.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null or it is not of type InstanceNameId.
- If resourceTemplateCol is null or empty or it is not of type ResourceTemplate.

FeatureException is thrown if any of the following situation occurs:

- if neInstanceNameId is not the InstanceNameId of default VDC.

Parameters

- opContext—Operational context
- neInstanceNameId—InstanceNameId of default VirtualNetworkElement
- resourceTemplateCol—List of new ResourceTemplate Objects.

Return Value

List of InstanceNameIds of ResourceTemplates that are persisted in the database.
createVdcs

Creates a VDC in the specified device.

FeatureException is thrown if any of the following situation occurs:

- if neInstanceNameId is not the InstanceNameId of default VDC.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameId is null or it is not of type InstanceNameId.
- If neInstanceNameId is not a valid VirtualNetworkElement InstanceNameId.

**Parameters**

opContext—Operational context

neInstanceNameId—InstanceNameId of the default VirtualNetworkElement

vdcCol—List of VirtualNetworkElement to be created

**Return Value**

Collection of VirtualNetworkElement that are going to be persisted in the database. The returned collection will have the following associations and the rest will be cleared:

- VdcResourceLimit corresponding to VirtualNetworkElement
- VirtualNetworkElementStatus corresponding to VirtualNetworkElement
- NetworkInterface present in the VirtualNetworkElement
- AbstractNetworkElementStatus present in the VirtualNetworkElement
- SshServerSetting corresponding to the VirtualNetworkElement

createVdcsWithResourceTemplate

Creates VDCs along with the resource limits. Given an InstanceNameId of ResourceTemplate, corresponding VdcResourceLimit will be created along with VDC.

ValidationException is thrown if any of the following situation occurs:

- If resourceTemplateId is null or it is not of type InstanceNameId.
- Only existing ResourceTemplates (exist in database) can be used to create resourceLimits.
- If vdcCol is null or empty or it is not of type VirtualNetworkElement.

**Parameters**

opContext—Operational context

resourceTemplateId—InstanceNameId of resourceTemplate to be configured along with VDC.

vdcCol—List of VirtualNetworkElement to be created.

**Return Value**

List of InstanceNameId of VDCs that are persistent in the database.
deleteResourceTemplates

Deletes one or more resource templates configured in the device.
ValidationException is thrown if any of the following situation occurs:
- If resourceTemplateCol is null or empty or it is not of type InstanceNameId.
- Only existing ResourceTemplates (exist in database) can be modified.

Parameters
opContext—Operational context
resourceTemplateCol—InstanceNameId of ResourceTemplates that are to be deleted.

Return Value
void

deleteVdcs

Deletes one or more VDCs from the device.
ValidationException is thrown if any of the following situation occurs:
- If vdcInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- Only existing VirtualNetworkElement(exist in database) can be deleted.

Parameters
opContext—Operational context
vdcInstanceNameIdCol—List of InstanceNameId of VirtualNetworkElement that are to be deleted.

Return Value
void

getAllVdcsWithChassis

Gets the list of virtual network elements for the list of specified chassis.

Parameters
opContext—Operational context
InstanceNameId—List of chassis ids

Return Value
List of VirtualNetworkElement
getResourceTemplatesInNetworkElement

Returns ResourceTemplates configured in the chassis. Given the instanceNameId of the default VDC. The API returns the collection of resource templates configured in the device.

FeatureException is thrown if any of the following situation occurs:
- if neInstanceNameId is not the InstanceNameId of default VDC.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameId is null or it is not of type InstanceNameId.
- If neInstanceNameId is not a valid VirtualNetworkElement neInstanceNameId.

Parameters

opContext—Operational context
neInstanceNameId—InstanceNameId of the default VirtualNetworkElement

Return Value

List of ResourceTemplate configured in the default VDC. The returned Collection will contain the following associations and the rest will be cleared:
- TemplateResourceLimit corresponding to ResourceTemplate
- VirtualNetworkElement corresponding to ResourceTemplate
- ResourceTemplate present in the TemplateResourceLimit

getVdcs

Returns VDC's corresponding to their instanceNameId's.

ValidationException is thrown if any of the following situation occurs:
- If vdcInstanceNameIdCol is null or empty or it is not of type InstanceNameId.
- If vdcInstanceNameIdCol is not a valid InstanceNameId of AbstractNetworkElement object.

Parameters

opContext—Operational context
vdcInstanceNameIdCol—InstanceNameId's of VirtualNetworkElement

Return Value

List of VirtualNetworkElement. The returned collection will have the following associations and the rest will be cleared.
- VdcResourceLimit corresponding to VirtualNetworkElement
- VirtualNetworkElementStatus corresponding to VirtualNetworkElement
- NetworkInterface present in the VirtualNetworkElement
- AbstractNetworkElementStatus present in the VirtualNetworkElement
- SshServerSetting corresponding to the VirtualNetworkElement
getVdcInChassis

Returns the collection of VirtualNetworkElement for the given chassis. ValidationException is thrown if any of the following situation occurs:

- If chassisId is null or it is not of type InstanceNameId.
- If chassisId is not a valid Chassis InstanceNameId.

Parameters

opContext—Operational context
neInstanceNameId—InstanceNameId of Chassis.

Return Value
List of VirtualNetworkElement. The returned collection will have the following associations and the rest will be cleared.

- VdcResourceLimit corresponding to VirtualNetworkElement
- VirtualNetworkElementStatus corresponding to VirtualNetworkElement
- NetworkInterface present in the VirtualNetworkElement
- AbstractNetworkElementStatus present in the VirtualNetworkElement
- SshServerSetting corresponding to the VirtualNetworkElement

modifyResourceTemplates

Modifies Resource Templates that are configured in the device. ValidationException is thrown if any of the following situation occurs:

- If resourceTemplateCol is null or empty or it is not of type ResourceTemplate.
- Only existing ResourceTemplate (exist in database) can be modified.

Parameters

opContext—Operational context
resourceTemplateCol—List of modified ResourceTemplate objects.

Return Value
void

modifyVdcs

Modifies a list of configured VDC's. ValidationException is thrown if any of the following situation occurs:

- If vneCol is null or empty or it is not of type VirtualNetworkElement.
- Only existing VirtualNetworkElement(exist in database) can be modified.
unbindVdcsWithResourceTemplate

Removes a specified VdcResourceLimit from the VirtualNetworkElement. ValidationException is thrown if any of the following situation occurs:

- If vneId is null or empty or it is not of type InstanceNameId.
- If vdcResourceId is null or empty or it is not of type InstanceNameId.
- Only existing VdcResourceLimit (exist in database) can be removed.

Parameters

opContext—Operational context
vdcResourceId—InstanceNameId of VdcResourceLimit to be configured along with VDC.
vneId—InstanceNameId of VDC from which the corresponding VdcResourceLimit has to be removed.

Return Value

void
VlanApp Service

This chapter describes the DCNM web services’ API methods for the VlanApp service.

Information About VlanApp Service

VLANs allow you to partition a LAN into smaller logical broadcast domains. Devices in a VLAN will not receive broadcasts packets from other VLANs even if they are connected to the same physical LAN segment.

The private VLAN feature allows you to segment a VLAN into smaller broadcast domains that share the same subnet. For example, you can partition a VLAN into a primary VLAN and one or more secondary VLANs.

addAllowedVlansOnTrunks

Add a range of VLAN as allowed VLAN on the given ports irrespective of port mode. If the specified VLAN does not exist, then an Incomplete Normal VLAN will be created and bound to the port.

ValidationException is thrown if any of the following situation occurs:

- If portNameIds is null or empty.
- If the elements in the portNameIds is not valid InstanceNameId of SwitchedNetworkInterface.
- If vlanRange is null or empty.
- If the IDs in the vlanRange are out of valid Range.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in portNameIds does not exist.

Parameters

- opContext—Operational context.
- portNameIds—List of InstanceNameId of the port for which the allowed VLANs are to be added.
- vlanRange—IntegerRange of VLAN IDs to be added as allowed VLANs for the given port (e.g. 1-10, 20, 35, 50-120).

Return Value

void
bindAccessPortsToVlan

Binds the given ports to the specified VLAN. If the specified VLAN does not, an incomplete Normal VLAN will be created.

Following associations will be updated between the Port and VLAN:

- SwitchedNetworkInterfaceBelongsToVlan with association type as access

ValidationException is thrown if any of the following situation occurs:

- If portNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in portNameIds is not a valid InstanceNameId of SwitchedNetworkInterface object.
- If vlanId is null.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in portNameIds does not exist.

Parameters

- opContext—Operational context.
- portNameIds—List of InstanceNameId of SwitchedNetworkInterfaces which has to bound to the specified VLAN.
- vlanId—Integer VLAN ID of the VlanExternal to which the given access ports have to be bound.

Return Value

List of SwitchedNetworkInterfaceBelongsToVlan objects.

bindHostPortsToPrivateVlan

Binds PVLAN host ports to a secondary VLAN. If the specified VLAN does not, an incomplete Normal VLANVlanExternal will be created.

Following associations will be updated between the Port and VLAN:

- SwitchedNetworkInterfaceBelongsToVlan with association type as PVLAN_HOST_PRIMARY for primary VLAN and PVLAN_HOST_SECONDARY for the given secondary VLAN will be created.

ValidationException is thrown if any of the following situation occurs:

- If portNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in portNameIds is not a valid InstanceNameId of SwitchedNetworkInterface object.
- If corresponding object for the elements in portNameIds does not exist.
- The given ports should be in Host mode.
- If secondaryVlanNameId or primaryVlanId is null.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in portNameIds does not exist.

Parameters

- opContext—Operational context.
- primaryVlanId—VLAN ID of the PrimaryVlan to be bound to the given ports.
Send document comments to nexus7k-docfeedback@cisco.com

secondaryVlanId—VLAN ID of the SecondaryVlan to be bound to the given ports.
secondaryHostPortNameIdCol—List of InstanceNameId of SwitchedNetworkInterfaces in Host Mode.

Return Value
List of SwitchedNetworkInterfaceBelongsToVlan objects.

bindNativeVlanOnTrunks

Set the specified VLAN as native VLAN for the given ports. If the specified VLAN does not exist, then an Incomplete Normal VLAN will be created and bound to the ports.

ValidationException is thrown if any of the following situation occurs:
• If portNameIds is null or empty.
• If the elements in the portNameIds is not valid InstanceNameId of SwitchedNetworkInterface
• If vlanRange is null or empty.
• If the IDs in the vlanRange are out of valid Range.

IntegrityException is thrown if any of the following situation occurs:
• If the corresponding object for InstanceNameIds in portNameIds does not exist.

Parameters
opContext—Operational context.
portNameIds—List of InstanceNameIds of SwitchedNetworkInterface ports.
vlanId—Integer ID of the VLAN to be set as native VLAN for the given ports.

Return Value
void

bindPromiscuousPortsToPrivateVlan

Binds Promiscuous ports to the given primary VLAN and the given secondary VLANs will also be mapped to the created promiscuous association. If the specified VLAN does not exist, an incomplete Normal VLAN will be created.

Following associations will be updated between the Port, primary VLAN and secondary VLANs:
• PromiscuousPvlanMapping

ValidationException is thrown if any of the following situation occurs:
• If portNameIds is null or empty or the elements in the list are not of type InstanceNameId.
• If elements in portNameIds is not a valid InstanceNameId of SwitchedNetworkInterface object.
• If primaryVlanId is null or not InstanceNameId of PrimaryVlan or corresponding object does not exist.
• If secondaryVlanIds is null or empty.
• If any of the VLANIds in secondaryVlanIds overlaps with primaryVlanId.

IntegrityException is thrown if any of the following situation occurs:
bindSecondaryVlansToPrimary

Binds the given secondary VLANs to the specified primary VLAN.

If the specified secondary VLAN object does not exist, incomplete secondary VLAN object will be created and bound.

ValidationException is thrown if any of the following situation occurs:

- If primaryVlanNameId is null or not InstanceNameId of primary VLAN.
- In case of Catalyst 6500 series switches, if the VTP Mode is in client / server.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for primaryVlanNameId does not exist.

Parameters

opContext—Operational context.

primaryVlanNameId—InstanceNameId of the primary VLAN.

secondaryVlanRange—IntegerRange of Secondary VLAN IDs to be bound with the given primary VLAN (e.g. 1-10,20,35,50-120).

Return Value

void

bindSecondaryVlansToVlanNetworkInterface

Binds a set of secondary VLANs to an VlanNetworkInterface (SVI).

ValidationException is thrown if any of the following situation occurs:

- If vlanNetworkInterfaceNameId is null or not valid InstanceNameId of VlanNetworkInterface.
- If secondaryVlanId is null or the list is empty.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding VlanNetworkInterface object for the vlanNetworkInterfaceNameId does not exist.
createSecondaryVlans

Create a list of secondary VLANs for a given primary InstanceNameId. A primary VLAN must exist to create a secondary VLAN.

ValidationException is thrown if any of the following situation occurs:

- If secVlanCol is null or empty.
- If elements in secVlanCol is not a valid type of SecondaryVlan.
- If the secondaryVlanType is null.
- In case of Catalyst 6500 series switches, if the VTP Mode is in client / server.
- If the VLAN name is not unique in the device.
- If the specified VLAN ID overlap with VLAN ID of an Internal VLAN.
- If the VLAN is a default VLAN, e.g.: VLAN 1, VLAN 1002 in Catalyst 6500 series switches.
- If the specified primary VLAN have more than 1 Isolated VLAN mapped to it

IntegrityException is thrown if any of the following situation occurs:

- If a VLAN with same vlanId in the specified device already exist.
- If the corresponding object for primaryVlanNameId does not exist.

Parameters

- opContext—Operational context.
- primaryVlanNameId—PrimaryVlan object for which secondary VLAN have to be associated.
- secVlanCol—List of SecondaryVlan object which need to be created and associated to a primary VLAN.

Return Value

List of InstanceNameId of the newly created SecondaryVlan.

createVlanInNetwork

Create a normal VLAN on a set of network elements. If network elements are not specified, the VLAN will be created on all the managed network elements.

ValidationException is thrown if any of the following situation occurs:

- If vlanId is null.
createVlans

Create a list of VLANs in a network elements. This method can be used to create normal as well as private VLANs. None of the associations like (SVI, secondary VLAN) will be considered in this API.

Following associations are to be provided for the given VLAN:
- Network Element reference

ValidationException is thrown if any of the following situation occurs:
- If vlanCol is null or empty.
- If elements in vlanCol is not a valid type of VlanExternal or with AbstractNetworkElement reference not specified.
- If the VLAN name is not unique in the device.
- If the specified VLAN ID overlap with VLAN ID of an Internal VLAN.
- In case of Catalyst 6500 series switches, if the VTP Mode is in client / server.
- If the VLAN is a default VLAN. E.g.: VLAN 1, VLAN 1002(In case of Catalyst 6500 series switches) etc.

IntegrityException is thrown if any of the following situation occurs:
- If a VLAN with same vlanId in the specified device already exist.
- If the corresponding object to the specified AbstractNetworkElement does not exist.
deleteAllowedVlansOnTrunks

Delete a range of VLANs from the allowed-VLAN list of trunk ports. The VLAN can be of any type. If all the VLANs are removed from the allowed VLANs list for any port, allowed VLANs will be set to default (1-4094).

Following validations will be done:
- The given VLAN should be a Normal VLAN only.
- The given port should be a Trunk port only.

ValidationException is thrown if any of the following situation occurs:
- If portNameIds is null or empty.
- If the elements in the portNameIds is not a valid InstanceNameId of SwitchedNetworkInterface.
- If vlanRange is null or empty.
- If the IDs in the vlanRange are out of valid Range.

IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for InstanceNameIds in portNameIds does not exist.

Parameters

- opContext—Operational context.
- portNameIds—List of InstanceNameIds of the port from which the allowed VLANs are to be deleted.
- vlanRange—IntegerRange of VLAN IDs to be deleted from the allowed VLANs list for the given port (e.g. 1-10,20,35,50-120).

Return Value

void

deleteVlans

Deletes a collection of VLANs. If the VLAN is a primary VLAN, all the associated secondary VLANs will be deleted. If the specified VLAN have associations, then the VLAN will be set to Incomplete state and its associations are preserved.

ValidationException is thrown if any of the following situation occurs:
- If vlanNameIds is null or empty.
- If the elements in the vlanNameIds is not a InstanceNameId of VLAN
- If the specified VLAN is an internal or default VLAN.

Parameters

- opContext—Operational context.
- vlanNameIds—List of InstanceNameIds of VLANs to be deleted.

Return Value

void

Parameters

- opContext—Operational context.
- vlanCol—List of VlanExternal objects to be created.

Return Value

List of InstanceNameId corresponding to the newly created VLANs.
deleteVlansInNetwork

Deletes a collection of VLANs from a list of network elements. If the VLAN is a primary VLAN, all the associated secondary VLANs will be unbound. Also if the primary VLAN has any incomplete Secondary VLANs bound to it, then the incomplete VLANs will be deleted.

ValidationException is thrown if any of the following situation occurs:

- If vlanIds is null or empty.
- If the specified VLAN is an internal or defaultVLAN.
- In case of Catalyst 6500 series switches, if the VTP Mode is in client / server.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding AbstractNetworkElement objects for the InstanceNameId in the neNameIds does not exist.

Parameters

- opContext—Operational context.
- vlanNameIds—List of InstanceNameId of the VLAN to be deleted.

Return Value

void

disablePrivateVlanService

Disables Private VLAN Service in the network element. Service Enabling/Disabling is supported in DCOS platform. If this API is called with the network elements of Catalyst 6500 series switches then FeatureException will be thrown.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null.
- If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
enablePrivateVlanService

Enables private VLAN service for the network element. Service Enabling/Disabling is supported in DCOS platform. If this API is called with the network elements of Catalyst 6500 series switches then FeatureException will be thrown.

ValidationException is thrown if any of the following situation occurs:
- If neInstanceNameIdCol is null.
- If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters
- opContext—Operational context.
- neInstanceNameIdCol—Collection of InstanceNameId of the network elements.

Return Value
- void

getActivePortsCountInVlans

Returns List of number of active ports present for given VLAN InstanceNameId list. Count will be done for the ports based on the VLAN type:
- Normal VLAN—number of active access ports.
- Primary VLAN—number of active promiscuous ports.
- Secondary VLAN—number of active PVLAN host ports.

ValidationException is thrown if any of the following situation occurs:
- If vlanNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in vlanNameIds is not a valid InstanceNameId of VlanExternal object.

IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for InstanceNameIds in vlanNameIds does not exist.

Parameters
- opContext—Operational context.
getInactivePortsCountInVlans

Returns List of number of inactive ports present for given VLAN InstanceNameId list. Count will be done for the ports based on the VLAN type:

- Normal VLAN—number of inactive access ports.
- Primary VLAN—number of inactive promiscuous ports.
- Secondary VLAN—number of inactive PVLAN host ports.

ValidationException is thrown if any of the following situation occurs:
- If vlanNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in vlanNameIds is not a valid InstanceNameId of VlanExternal object.

IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for InstanceNameIds in vlanNameIds does not exist.

Parameters
- opContext—Operational context.
- vlanNameIds—List of InstanceNameId of the VLAN objects for which the Inactive port counts are to be queried.

Return Value
List of Inactive port counts.

getInternalVlansInNetworkElement

Returns Internal VLAN objects corresponding to the network element. Associations will not be available for the VlanInternal Objects.

ValidationException is thrown if any of the following situation occurs:
- If neNameId is null or not an InstanceNameId of AbstractNetworkElement.

IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for neNameId does not exist.

Parameters
- opContext—Operational context.
- neNameId—InstanceNameId of network element for which Internal VLANS are to be queried.
getPortsCountInVlans

Returns List of number of ports present for given VLAN InstanceNameId list. Count will be done for the ports based on the VLAN type:
- Normal VLAN—number of access ports with given status.
- Primary VLAN—number of promiscuous ports with given status.
- Secondary VLAN—number of PVLAN host ports with given status.

ValidationException is thrown if any of the following situation occurs:
- If vlanNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in vlanNameIds is not a valid InstanceNameId of VlanExternal object.

IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for InstanceNameIds in vlanNameIds does not exist.

Parameters
opContext—Operational context.
vlanNameIds—List of InstanceNameId of the VLAN objects for which the port counts based on the NetworkInterfaceOperationStatus are to be queried.
status—NetworkInterfaceOperationStatus. If null, returns number of ports in the VLAN irrespective of operational status.

Return Value
List of port counts

getPortsInVlan

Returns a list of ports belonging to a VLAN. Returns a collection of SwitchedNetworkInterface (Physical Ports and port channel) based on the VLAN type provided:
- Normal VLAN—Access ports
- Primary VLAN—Promiscuous ports
- Secondary VLAN—Private VLAN host ports

Following associations will be available for each NetworkInterface:
- NetworkInterfaceName.
- NetworkInterfaceStatus.
- PromiscuousPvlanMapping. Secondary VLANs collection will be populated for Promiscuous PVLAN mapping

ValidationException is thrown if any of the following situation occurs:
- If vlanNameId is null or not a valid InstanceNameId of VlanExternal.
getPrivateVlanServiceState

Returns list of boolean stating the corresponding PVLAN service is enabled or not.

ValidationException is thrown if any of the following situation occurs:

- If neInstanceNameIdCol is null.
- If neInstanceNameIdCol does not contain a valid network element InstanceNameId.
- If the network element does not exist in the database.

Parameters

opContext—Operational context.
neInstanceNameIdCol—Collection of InstanceNameId of the network elements.

Return Value

void

g etPrivateVlansInNetworkElement

Returns private VLANs configured in a network element. Given the InstanceNameId of the network element, returns a List of primary VLANs and their corresponding secondary VLANs.

Following associations will be available for each VLAN:

- VLAN status
- Secondary VLANs collection incase of primary VLAN. This secondary VLAN references VlanStatus
- Primary VLAN reference incase of Secondary VLAN. This primary VLAN references VlanStatus

ValidationException is thrown if any of the following situation occurs:

- If neNameId is not InstanceNameId of AbstractNetworkElement.
IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for neNameId does not exist.

Parameters

opContext—Operational context.
**getSecondaryVlans**

Returns secondary VLANs of one or more primary VLANs.

Following associations will be available for each VLAN:

- VLAN status

ValidationException is thrown if any of the following situation occurs:

- If pvlanNameIds is null or empty or it is not of type InstanceNameId.
- If elements in pvlanNameIds is not a valid InstanceNameId of PrimaryVlan object.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in pvlanNameIds does not exist.

**Parameters**

- opContext—Operational context.
- pvlanNameIds—List of PrimaryVlan InstanceNameIds whose secondary VLANs has to be queried.

**Return Value**

List of SecondaryVlan objects with its associated PrimaryVlan reference.

---

**getTrunksForVlan**

Returns trunk ports allowing a VLAN.

Returns a List of SwitchedNetworkInterface based on the VLAN type:

- Normal VLAN—Trunk ports which allow the VLAN

Following associations will be available for each NetworkInterface:

- NetworkInterfaceName
- NetworkInterfaceStatus

ValidationException is thrown if any of the following situation occurs:

- If vlanNameId is null or not a valid InstanceNameId of VlanExternal.
- If the given VLAN is private VLANs (PrimaryVlan or SecondaryVlan).

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for vlanNameId does not exist.

**Parameters**

- opContext—Operational context.
- vlanNameId—InstanceNameId of VlanExternal from which the ports are to be queried.
getVlanGlobalSettings

Returns a list of VLAN global settings for the given network elements. ValidationException is thrown if any of the following situation occurs:

- If neNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in the neNameIds are not of InstanceNameId type of AbstractNetworkElement.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in neNameIds does not exist.

Parameters

- opContext—Operational context.
- neNameIds—InstanceNameId of network elements whose VLAN IDs are to be queried. If null, will return all VLAN IDs from all the managed network elements.

Return Value

List of VlanGlobalSetting objects corresponding to the given network elements.

getVlanIdsInNetwork

Returns IDs of VLANs configured on a set of network elements. This API will return IDs of normal and primary VLANs. No associations will be populated in this API, only VLAN ID range will be returned by this API

ValidationException is thrown if any of the following situation occurs:

- If elements in the neNameIds are not of InstanceNameId type of AbstractNetworkElement.

Parameters

- opContext—Operational context.
- neNameIds—InstanceNameId of network elements whose VLAN IDs are to be queried. If null, will return all VLAN IDs from all the managed network elements.

Return Value

Range of VLAN IDs IntegerRange

getVlanNetworkInterfaces

Returns a VlanNetworkInterface (SVI) objects corresponding to a list of VLANs

Following associations will be available for each interface:

- Name.
- Interface Status.
Secondary VLAN collection. ValidationException is thrown if any of the following situation occurs:

- If vlanNameIds is null or empty or the elements in the list are not of the type InstanceNameId.
- If elements in vlanNameIds is not a valid InstanceNameId of VlanExternal object.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in vlanNameId does not exist.

**Parameters**

- opContext—Operational context.
- vlanNameIds—List of InstanceNameId of VlanExternal objects.

**Return Value**

List of corresponding VlanNetworkInterface objects for the given VLAN InstanceNameId.

### getVlans

Returns corresponding VLAN objects for the given VLAN InstanceNameId list. Only VLAN external objects is allowed in this API.

Following associations will be available for each VLAN:

- VLAN status
- Secondary VLANs collection incase of primary VLAN. This secondary VLAN references VlanStatus
- Primary VLAN reference incase of Secondary VLAN. This primary VLAN references VlanStatus

ValidationException is thrown if any of the following situation occurs:

- If vlanNameIds is null or empty or it is not of type InstanceNameId.
- If elements in pvlanNameIds is not a valid InstanceNameId of VlanExternal object.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in vlanNameIds does not exist.

**Parameters**

- opContext—Operational context.
- vlanNameIds—InstanceNameIds of VLANs.

**Return Value**

List of VlanExternal objects corresponding to the specified InstanceNameId list.

### getVlansInNetwork

Returns VLAN objects corresponding to a list of VLAN IDs for a list of network elements. If the VLAN IDs passed is null then the all VLANs are returned for the given network elements. If null is passed for network element InstanceNameId list then VLANs in all the managed network elements are returned.
Following associations will be available for each VLAN:

- VLAN status
- PVRST Settings
- MST Setting
- Secondary VLANs collection incase of primary VLAN. This secondary VLAN references to VlanStatus and STP settings
- Primary VLAN reference incase of Secondary VLAN. This primary VLAN references VlanStatus and STP settings

ValidationException is thrown if any of the following situation occurs:

- If elements in the neNameIds are not of InstanceNameId type of AbstractNetworkElement.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in neNameIds does not exist.

**Parameters**

opContext—Operational context.

vlanIdRange—Range of VLAN IDs (e.g. 1-10,20,35,50-120). If null, returns all VLANs.

neNameIds—InstanceNameId list of network elements for which VLAN IDs are to be queried. If null, returns VLANs on all managed network elements.

**Return Value**

Ordered list of VlanExternal objects.

---

Returns VLANs configured in a network element. Given the InstanceNameId of the network element, returns a list of following types of VLAN:

- Normal VLAN.
- Private VLAN (Primary and Secondary).

This API will not return any internal VLANs.

Following associations will be available for each VLAN:

- VLAN status
- Secondary VLANs collection incase of primary VLAN. This secondary VLAN references to VlanStatus
- Primary VLAN reference incase of Secondary VLAN. This primary VLAN references VlanStatus

ValidationException is thrown if any of the following situation occurs:

- If neNameId is not InstanceNameId of AbstractNetworkElement.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for neNameId does not exist.

**Parameters**

opContext—Operational context.
getVlansWithSecurityAssociations

Return the security associations corresponding to VLANs. Following associations will be populated for each VLAN:

- DaiSetting
- DhcpSnoopingSetting

ValidationException is thrown if any of the following situation occurs:

- If vlanNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in vlanNameIds is not a valid InstanceNameId of VlanExternal object.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in vlanNameIds does not exist.

Parameters

- opContext—Operational context.
- vlanNameIds—InstanceNameIds of VLANs.

Return Value

List of VlanExternal objects with security associations corresponding to the specified InstanceNameId.

getVlansWithSpanAssociations

Return the SPAN associations corresponding to VLANs. Following associations will be populated for each VLAN:

- Local SPAN to source VLAN
- Filtered VLAN

Following objects will be populated on other end of LocalSpanSessionAppliesToVlan/filtered VLAN association:

- LocalSpanSession

ValidationException is thrown if any of the following situation occurs:

- If vlanNameIds is null or empty or it is not of type InstanceNameId.
- If elements in vlanNameIds is not a valid InstanceNameId of VlanExternal object.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in vlanNameIds does not exist.

Parameters

- opContext—Operational context.
getVtpGlobalSettings

Returns a list of VTP global settings for the given network elements. VtpGlobalSetting object will be null in Nexus 7000 series switch platform.

ValidationException is thrown if any of the following situation occurs:
- If neNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in the neNameIds are not of InstanceNameId type of AbstractNetworkElement.

IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for InstanceNameIds in neNameIds does not exist.

Parameters
opContext—Operational context.
neNameIds—InstanceNameId of network elements whose VLAN IDs are to be queried. If null, will return all VLAN IDs from all the managed network elements.

Return Value
List of VtpGlobalSetting objects corresponding to the given network elements.

modifyAllowedVlansOnTrunks

Modify the existing allowed VLAN for the list of given ports irrespective of port mode. The old associations will be replaced with the given VLAN IDs. If the specified VLAN does not exist, then an Incomplete Normal VLAN will be created and bound to the port.

ValidationException is thrown if any of the following situation occurs:
- If portNameIds is null or empty.
- If the elements in the portNameIds is not valid InstanceNameId of SwitchedNetworkInterface
- If vlanRange is null or empty.
- If the IDs in the vlanRange are out of valid Range.

IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for InstanceNameIds in portNameIds does not exist.

Parameters
opContext—Operational context.
portNameIds—List of InstanceNameIds of the port for which the allowed VLANs are to be modified.
vlanRange—IntegerRange of VLAN IDs to be modified as allowed VLANs for the given port (e.g. 1-10,20,35,50-120).
modifySecondaryVlansForPromiscuousPorts

Modifies the existing secondary VLANs associated with the existing promiscuous PVLAN association for the given list of ports. If the specified VLAN does not exist, an incomplete normal VLAN will be created.

Following associations will be replaced between the Port and secondary VLANs:

- PromiscuousPvlanMapping.

ValidationException is thrown if any of the following situation occurs:

- If portNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in portNameIds is not a valid InstanceNameId of SwitchedNetworkInterface object.
- If primaryVlanId is null or not InstanceNameId of PrimaryVlan or corresponding object does not exist.
- If secondaryVlanIdRange is null or empty.
- If any of the VLANIds in secondaryVlanIdsRange overlaps with primaryVlanId.
- If promiscuous binding does not exist for the given ports.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in portNameIds does not exist.

Parameters

- opContext—Operational context.
- portNameIdCol—List of InstanceNameId of SwitchedNetworkInterfaces.
- secondaryVlanIdRange—IntegerRange of secondary VLAN IDs (e.g. 1-10,20,35,50-120) to be mapped to the promiscuous port.

Return Value

List of PromiscuousPvlanMappings.

modifyVlanGlobalSetting

Modify the an existing VLAN global setting object with the given global setting. The old association will be replaced with the new global setting.

ValidationException is thrown if any of the following situation occurs:

- If neNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in the neNameIds are not of InstanceNameId type of AbstractNetworkElement.
- If setting is null.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in neNameIds does not exist.
modifyVlanTypeToNormal

Modify a list of VLAN’s type to Normal. This will return the list of type changed VLAN object that has been created in the server.

ValidationException is thrown if any of the following situation occurs:
- If vlanCol is null or empty or the elements in the list are not of type VlanExternal.
- If the elements in the list are of VlanExternal with the already exist in the device.
- In case of Catalyst 6500 series switches, if the VTP Mode is in client / server.

IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for objects in vlanCol does not exist.

Parameters
- opContext—Operational context.
- vlanCol—List of VLAN containing Normal VLAN and secondary VLANs

Return Value
List of modified (Normal) VLANs

modifyVlanTypeToPrimary

Modify a list of VLAN’s type to Primary. This will return the list of type changed VLAN object that has been created in the server.

ValidationException is thrown if any of the following situation occurs:
- If vlanCol is null or empty or the elements in the list are not of type VlanExternal.
- If the elements in the list are of VlanExternal with the already exist in the device.
- In case of Catalyst 6500 series switches, if the VTP Mode is in client / server.

IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for objects in vlanCol does not exist.

Parameters
- opContext—Operational context.
- vlanCol—List of VLAN containing Normal VLANs and secondary VLANs
modifyVlanTypeToSecondary

Modify a list of VLAN's type to Secondary. This will return the list of type changed VLAN object that has been created in the server.

ValidationException is thrown if any of the following situation occurs:

- If vlanCol is null or empty or the elements in the list are not of type VlanExternal.
- If the elements in the list are of SecondaryVlan with the same secondaryVlanType overlap with any of the VLAN already existing in the device.
- If secondaryVlanType is null.
- If the Isolated VLAN limit of the primary VLAN corresponding to given Secondary VLAN objects exceeded.
- In case of Catalyst 6500 series switches, if the VTP Mode is in client / server.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for objects in vlanCol does not exist.

Parameters

- opContext—Operational context.
- vlanCol—List of VLAN containing Normal VLANs and Primary VLANs
- secondaryVlanType—Secondary VLAN type (community/isolated)

Return Value

List of modified VLANs

modifyVlans

Modify a collection of existing VLAN objects. No associations will be updated. Only VLAN specific parameters will be updated.

ValidationException is thrown if any of the following situation occurs:

- If modifiedVlanCol is null or empty.
- If the elements in the modifiedVlanCol is not of type VlanExternal
- If the specified VLAN is an internal or default VLAN.
- If the private VLANs are Remote SPAN enabled.
- In case of Catalyst 6500 series switches, if the VTP Mode is in client / server.

PropertiesException is thrown if any of the following situation occurs:

- If any of the attributes for VlanExternal object modifiedVlanCol is not valid / not supported based on the platform type.

Example:
unbindAccessPortsFromVlan

Unbinds the access VLAN association for the given ports. After unbinding, the port will be bound to default VLAN.

Following associations will be updated between the Port and VLAN:

- `SwitchedNetworkInterfaceBelongsToVlan` with association type as access

ValidationException is thrown if any of the following situation occurs:

- If `portNameIds` is null or empty or the elements in the list are not of type `InstanceNameId`.
- If elements in `portNameIds` is not a valid `InstanceNameId` of `SwitchedNetworkInterface` object.
- If `vlanId` is null.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for `InstanceNameIds` in `portNameIds` does not exist.

Parameters

- `opContext`—Operational context.
- `portNameIds`—List of `InstanceNameId` of `SwitchedNetworkInterfaces` which has to unbound from the access VLAN.

Return Value

- `void`

unbindHostPortsFromPrivateVlan

Unbinds the PVLAN host association for the given ports.

Following associations will be updated between the Port and VLAN:

- `SwitchedNetworkInterfaceBelongsToVlan` with association type as access.

ValidationException is thrown if any of the following situation occurs:

- If `portNameIds` is null or empty or the elements in the list are not of type `InstanceNameId`.

Parameters

- `opContext`—Operational context.
- `portNameIds`—List of `InstanceNameId` of `SwitchedNetworkInterfaces` which has to unbound from the access VLAN.

Return Value

- `void`
unbindNativeVlanOnTrunks

Unbinds the trunk native VLAN association for the given ports. After unbinding, the port will be bound
to default VLAN.

Following associations will be updated between the Port and VLAN:

- SwitchedNetworkInterfaceBelongsToVlan with association type as access.

ValidationException is thrown if any of the following situation occurs:

- If portNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in portNameIds is not a valid InstanceNameId of SwitchedNetworkInterface object.
- If vlanId is null.

IntegrityException is thrown if any of the following situation occurs:

- If the corresponding object for InstanceNameIds in portNameIds does not exist.

Parameters

opContext—Operational context.

portNameIds—List of InstanceNameId of SwitchedNetworkInterfaces which has to unbound from the
PVLAN Host VLAN.

Return Value

void

unbindPromiscuousPortsFromPrivateVlan

Binds promiscuous ports to the given primary VLAN and the given secondary VLANs will also be
mapped to the created promiscuous association. If the specified VLAN does not, an incomplete normal
VLAN will be created.

Following associations will be updated between the Port, primary VLAN and secondary VLANs:

- PromiscuousPvlanMapping.

ValidationException is thrown if any of the following situation occurs:

- If portNameIds is null or empty or the elements in the list are not of type InstanceNameId.

Parameters

opContext—Operational context.

portNameIds—List of InstanceNameId of SwitchedNetworkInterfaces which has to unbound from the
trunk native VLAN.

Return Value

void
unbindSecondaryVlansFromPrimary

Unbinds the secondary VLANs from the primary VLAN if bound.
If the secondary VLAN is in Incomplete state, then the secondary VLAN will be deleted.
ValidationException is thrown if any of the following situation occurs:
- If secondaryVlanNameIds is null or empty or the elements in the list are not of type InstanceNameId.
- If elements in secondaryVlanNameIds is not a valid InstanceNameId of SecondaryVlan object.
- In case of Catalyst 6500 series switches, if the VTP Mode is in client/server.
IntegrityException is thrown if any of the following situation occurs:
- If the corresponding object for InstanceNameIds in secondaryVlanNameIds does not exist.

Parameters
- opContext—Operational context.
- secondaryVlanNameIds—List of InstanceNameId of secondary VLANs to be unbound from the Primary VLAN.

Return Value
- void

unbindSecondaryVlansFromVlanNetworkInterface

Unbinds the secondary VLANs from the given VlanNetworkInterface (SVI).
ValidationException is thrown if any of the following situation occurs:
- If vlanNetworkInterfaceNameId is null or not valid InstanceNameId of VlanNetworkInterface.
- If secondaryVlanIds is null or empty.
IntegrityException is thrown if any of the following situation occurs:
- If the corresponding VlanNetworkInterface object for the vlanNetworkInterfaceNameId does not exist.

Parameters
- opContext—Operational context.
- vlanNetworkInterfaceNameId—InstanceNameId of VlanNetworkInterface.
- secondaryVlanIds—List of InstanceNameId of secondary VLANs to be unbound from the VlanNetworkInterface.

Return Value
- void
Parameters

opContext—Operational context.

vlanNetworkInterfaceNameId—InstanceNameId of VlanNetworkInterface to which the Secondary VLANs are to be bound.

secondaryVlanIds—IntegerRange of secondary VLAN IDs (e.g. 1-10,20,35,50-120) to be bound to the VlanNetworkInterface.

Return Value

void
This chapter describes the DCNM web services’ API methods for the VpcApp service.

Information About VpcApp Service

A virtual port channel (vPC) enables links that are physically connected to two different devices (VDCs) to be viewed as a single logical port channel by the third device that forms the port channel with the vPC devices. You can set up a vPC between Layer 2 port channels.

The vPC is the port channel extended across two different chassis. With a vPC, an access layer switch could have the links of the same port channel terminating in two different distribution layer devices.

The vPC peer-link is a special inter switch link (ISL) that connects two Nexus 7000 series chassis. This link transfers control and data traffic across the chassis. The operational and configuration parameters of each chassis are exchanged on this link. The link is typically a port channel because the peer-link needs to be redundant and have enough bandwidth to transfer data traffic.

APIs are grouped as the following categories:

1. Query and Get APIs—Used to query data from the persisted database.
2. Create API—Used to create new vPCs.
3. Modify API—Used to modify vPCs, and vPC device-level configurations.
4. Delete APIs—Used to delete existing vPCs.
5. Enable and Disable APIs—Used to enable or disable a peer-link on port channel interfaces.
6. Synchronize APIs—Used to synchronize a primary vPC device with a secondary vPC device.

createVpc

Creates the given new Vpc and returns the instance name id of newly created Vpc.

The passed Vpc object should have Vpc etherchannel network interface endpoints and its neighbor endpoints and peer-link etherchannel network interface endpoints populated.

ValidationException is thrown if any of the following situation occurs:

- If newVpc is null.

PropertiesException is thrown if any of the following situation occurs:

- If Vpc number is not valid.
createVpcDomains

Creates Vpc Domain for the given list of domain Ids and returns the list of Vpc Domain instanceNameId's.

The size of device instanceNameId list and domain id list should be the same.

ParameterException is thrown if any of the following situation occurs:
- If Vpc Domain Ids collection size not equals to Device Ids collection.

FeatureException is thrown if any of the following situation occurs:
- If Vpc Domain already configured.

Parameters

opContext—Operational context
vpcDomainIds—List of Vpc domain objects to be created
eInstanceNameIds—List of Device Ids for which Vpc Domain to be created

Return Value
void

createVpcDomainsForIds

Creates Vpc Domain for the given list of domain Ids and returns the list of Vpc Domain instanceNameId's.

The size of device instanceNameId list and domain id list should be the same.

ParameterException is thrown if any of the following situations occurs:
- If Vpc Domain Ids collection size not equals to Device Ids collection.

FeatureException is thrown if any of the following situations occurs:
- If Vpc Domain already configured.

Parameters

opContext—Operational context
vpcDomainIds—List of Vpc domain objects to be created

Return Value
void
createVpcForSpecifiedEndpoints

Creates a Vpc between two Vpc end points with two peer-link end points.

ValidationException is thrown if any of the following situation occurs:
- If vpcNumber is null.
- If the vpcEndPoints is not a valid network interface InstanceNameId.
- If the peer-linkEndPoints contains one or more null element, or empty

PropertiesException is thrown if any of the following situation occurs:
- If vpcNumber is not valid Vpc number.
- If vpcEndPoints does not contain valid etherchannel InstanceNameIds.
- If peer-linkEndPoints does not contain valid etherchannel InstanceNameIds.

IntegrityException is thrown if any of the following situation occurs:
- If the Vpc with the given Vpc number already exist in the database.
- If the vpcEndPoints collection contains a NetworkInterface InstanceNameId that does not exist in the database.
- If a NetworkInterface in the peer-linkEndPoints contains duplicate NetworkInterface objects.

Parameters
opContext—Operational context
vpcNumber—Vpc number to be used for the Vpc
vpcEndPoints—List of InstanceNameId of Vpc endpoint
peer—-linkEndPoints the List of InstanceNameId of peer-link endpoint

Return Value
The InstanceNameId of the new Vpc

deleteVpcDomains

Deletes Vpc Domains for the list of instancename id's passed only if Vpc configuration does not exist

FeatureException is thrown if any of the following situation occurs:
- If vpcDomain has Vpc configuration

Parameters
opContext—Operational context
vpcDomainInstanceNameIds—array of InstanceNameId of Vpc Domain
deleteVpcs

Deletes all the given Vpc instances.
This method deletes all the given instances of VpcEndPoint
ValidationException is thrown if any of the following situation occurs:
• If vpcInstanceNameIds collection is null or it is empty.
• If vpcInstanceNameIds collection contains an element that is not of type VpcEndPoint
  InstanceNameId.
IntegrityException is thrown if the given vPCs do not exist in device.

Parameters
opContext—Operational context
vpcInstanceNameIds—array of InstanceNameId of Vpc

Return Value
void

deleteVpcsForVpcIds

Deletes the vPCs for the given vpcIds in the given network element.
This method removes given Vpc ids and Vpc association from corresponding
SwitchedEtherChannelNetworkItnerface instance in the given network element.
If the given vpcIds is null, then removes all Vpc ids in the given network element
ValidationException is thrown if any of the following situation occurs:
• If neInstanceNameId is null.
• If neInstanceNameId is not a valid Network Element InstanceNameId.
• If vpcIds is null.

Parameters
opContext—Operational context
InstanceNameId—InstanceNameId of NetworkElement
vpcIds—IntegerRange of vpcIds

Return Value
void
disableVpc

Disables vPC service on one or more network element. This API is applicable only for devices running DC-OS.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameIdCol is null.
- If the neInstanceNameIdCol contains one or more null element, or the collection is empty or it is not type InstanceNameId.

Parameters

- opContext—Operational context
- neInstanceNameIdCol—list of InstanceNameId of the Network Element for which the vPC should be disabled

Return Value

void

disableVpcForPortChannels

Deletes the vPCs for the given port channels.

This method removes vpcids and Vpc association from the given SwitchedEtherChannelNetworkInterface InstanceNameIds.

ValidationException is thrown if any of the following situation occurs:

- If portChannelInstanceNameIds collection is null or it is empty.
- If portChannelInstanceNameIds collection contains an element that is not of type SwitchedEtherChannelNetworkInterface InstanceNameId.

IntegrityException is thrown if the given portchannels do not exist in device.

Parameters

- opContext—Operational context
- portChannelInstanceNameIds—array of InstanceNameId of SwitchedEtherChannelNetworkInterface

Return Value

void

disableVpcPeerLinkForPortChannels

Disables Peer-link for the given port channels

This method disables peer-link for the given SwitchedEtherChannelNetworkInterface InstanceNameIds.

ValidationException is thrown if any of the following situation occurs:

- If portChannelInstanceNameIds collection is null or it is empty.
enableVpc

Enables vPC service on one or more network element. This API is applicable only for devices running DC-OS.

ValidationException is thrown if any of the following situation occurs:

- If the neInstanceNameIdCol is null.
- If the neInstanceNameIdCol contains one or more null element, or the collection is empty or it is not type InstanceNameId.

**Parameters**

- opContext—Operational context
- neInstanceNameIdCol—list of InstanceNameId of the Network Element for which the vPC should be enabled

**Return Value**

void

enableVpcPeerLinkForPortChannels

Enables Peer-link for the given port channels

This method enables peer-link for the given port channel InstanceNameIds and removes peer-link from existing port channel if it exists.

ValidationException is thrown if any of the following situation occurs:

- If portChannelInstanceNameIds collection is null or it is empty.
- If portChannelInstanceNameIds collection contains an element that is not of type SwitchedEtherChannelNetworkInterface InstanceNameId.

IntegrityException is thrown if the given portchannels do not exist in device.

**Parameters**

- opContext—Operational context
- portChannelInstanceNameIds—array of InstanceNameId of SwitchedEtherChannelNetworkInterface
getAllVpcs

Returns all the vPCs in the network.

For Each Vpc Id in the network, either Vpc Vpc or VpcEndPoint VpcEndPoint will be returned. Vpc is a network level instance for a Vpc Id(same) in two different devices, If there is no network level Vpc instance exist for a Vpc Id, then device level instance VpcEndPoint will be returned for that Vpc Id, If it exists, only Vpc instance will be returned instead of two VpcEndpoints. The vPCs returned by this method will have two Vpc end points, Each VpcEndPoint will be populated with corresponding Vpc port channel interface and peer-link port channel interface with their neighbor port channel interfaces.

Parameters

opContext—Operational context

Return Value

All the vPCs in network. The returned list will contain the list of VpcEndPoint instances and Vpc instances if it exists.

Following associations will be there for a Vpc: (other associations will be cleared)

1. Two Vpc end pointsVpcEndPoint of a Vpc
2. Each Vpc end point will have its Vpc port channel interface its neighbor port channel interfaces (SwitchedEtherChannelNetworkInterface)
3. Each Vpc end point will have its peer-link port channel interface with its neighbor port channel interface (SwitchedEtherChannelNetworkInterface)
4. Each Vpc end point will be populated with Vpc end point status
5. Each port channel interface will be populated with member ports
6. Each port channel interface will be populated with peer-link status

Following associations will be there for a VpcEndPoint: (other associations will be cleared)

1. Vpc end point will have its Vpc port channel interface its neighbor port channel interfaces (SwitchedEtherChannelNetworkInterface)
2. Each Vpc end point will have its peer-link port channel interface with its neighbor port channel interface (SwitchedEtherChannelNetworkInterface)
3. Each Vpc end point will be populated with Vpc end point status
4. Each port channel interface will be populated with member ports
5. Each port channel interface will be populated with peer-link status

getMultiChassisPortChannelsInNetworkElements

Returns multichassis vpc end points(port channel interface)for the given network elements.

Parameters

opContext—Operational context
getVpcDomainsInNetworkElements

Returns the Vpc domain of the given network elements.

The Vpc domain objects will be ordered in the returned List based on the order of given network element instance ids.

ValidationException is thrown if the argument passed is null or it is not a valid network element InstanceNameIds.

Parameters
opContext—Operational context
neInstanceNameIds—InstanceNameId of the network element instances whose Vpc domain information is required

Return Value
Ordered list of Vpc domain of the given network elements.

Following associations will be there for each VpcDomain: (other associations will be cleared)
1. VpcDomainSetting
2. VpcPeerKeepAliveStatus
3. VpcRoleStatus

getVpcIdsInNetworkElements

Returns all the Vpc ids in the given network element.

The Vpc ids ranges will be ordered in the returned List based on the order of given network element instance ids.

Parameters
opContext—Operational context
neInstanceNameIds—List of InstanceNameId of the network element for which the Vpc ids are required

Return Value
All the Vpc ids as List of IntegerRange in the given network elements.
**getVpcPeerLinkEndPoints**

Returns peer-link end points for the given network elements.

**Parameters**

- **opContext**—Operational context
- **neInstanceNameIds**—List of InstanceNameId of the network element for which the peer-link end points are required

  Peer-link end points will be populated with its neighbor endpoint (SwitchedEtherChannelNetworkInterface)

  Each end point will be populated with member port and its link associations

**Return Value**

The peer-link end points in the given network elements.

**getVpcStateOfNetworkElements**

Returns state of vPC service whether vPC is enabled or disabled in a list of network elements. Given the list of instance name IDs of the network elements, returns the list of Boolean values.

**ValidationException** is thrown if any of the following situation occurs:

- If `neInstanceNameIdCol` collection contains an element that is null or the collection is empty or it is not type `InstanceNameId`
- if the argument passed is null or it is not a valid network element `InstanceNameId`.

**Parameters**

- **opContext**—Operational context
- **neInstanceNameIdCol**—`InstanceNameId` of the one or more Network Element for which the AAA state is required

**Return Value**

The returned list will contain Boolean instances.

Boolean value `TRUE` represents vPC is enabled in the given network element.

Boolean value `FALSE` represents vPC is disabled in the given network element.

**getVpcs**

Returns the VpcEndPoint instances for the given Vpc end point InstanceNameIds.

If the given instance name id belongs to a Vpc, then the vPCs returned by this method will have two Vpc end points. Each VpcEndPoint will be populated with corresponding Vpc port channel interface and peer-link port channel interface with their neighbor port channel interfaces.

If the given instance name id belongs to a VpcEndPoint, then the VpcEndPoints returned by this method will be populated with corresponding Vpc port channel interface and peer-link port channel interface with their neighbor port channel interfaces.
getVpcsForIds

Returns the Vpc or VpcEndPoint instances for the given Vpc Ids.

For the given Vpc Id, either Vpc Vpc or VpcEndPoint VpcEndPoint will be returned. Vpc is a network level instance for a Vpc Id(same) in two different devices, If there is no network level Vpc instance exist for a Vpc Id, then device level instance VpcEndPoint will be returned for that Vpc Id. If it exists, only Vpc instance will be returned instead of two VpcEndPoints. The vPCs returned by this method will have two Vpc end points, Each VpcEndPoint will be populated with corresponding Vpc port channel interface and peer-link port channel interface with their neighbor port channel interfaces.

Parameters
- opContext—Operational context
- vpcIds—Vpc id for which the vPCs instances are required

Return Value
The vPCs or VpcEndPoints for the given Vpc Ids InstanceNameIds. The returned list will contain the list of VpcEndPoint instances.
getVpcsInNetworkElements

Returns all the vPCs in the given network elements.

For Each Vpc Id in the given network elements, either Vpc Vpc or VpcEndPoint VpcEndPoint will be returned. Vpc is a network level instance for a Vpc Id(same) in two different devices, If there is no network level Vpc instance exist for a Vpc Id, then device level instance VpcEndPoint will be returned for that Vpc Id. If it exists, only Vpc instance will be returned instead of two VpcEndPoints. The vPCs returned by this method will have two Vpc end points, Each VpcEndPoint will be populated with corresponding Vpc port channel interface and peer-link port channel interface with their neighbor port channel interfaces.

Parameters
opContext—Operational context
neInstanceNameId—InstanceNameId of the network element for which the vPCs are required

Return Value
All the vPCs in given network elements. The returned list will contain the list of VpcEndPoint instances and Vpc instances if it exists.

Following associations will be there for a Vpc: (other associations will be cleared)
1. Two Vpc end points VpcEndPoint of a Vpc
2. Each Vpc end point will have its Vpc port channel interface its neighbor port channel interfaces (SwitchedEtherChannelNetworkInterface)
3. Each Vpc end point will have its peer-link port channel interface with its neighbor port channel interface (SwitchedEtherChannelNetworkInterface)
4. Each Vpc end point will be populated with Vpc end point status
5. Each port channel interface will be populated with member ports
6. Each port channel interface will be populated with peer-link status

Following associations will be there for a VpcEndPoint: (other associations will be cleared)
1. Vpc end point will have its Vpc port channel interface its neighbor port channel interfaces (SwitchedEtherChannelNetworkInterface)
2. Each Vpc end point will have its peer-link port channel interface with its neighbor port channel interface (SwitchedEtherChannelNetworkInterface)
3. Each Vpc end point will be populated with Vpc end point status
4. Each port channel interface will be populated with member ports
5. Each port channel interface will be populated with peer-link status
modifyVpcDomains

Modifies any configuration changes for the given the List Vpc domain.
This method updates Vpc global settings.
ValidationException is thrown if any of the following situation occurs:
- if the argument passed is null or empty
FeatureException is thrown if any of the following situation occurs:
- if destination Ip address isnull

Parameters
opContext—Operational context
vpcDomains—List of Vpc domain objects to be set

Return Value
void

modifyVpcIdForPortChannel

Modifies the Vpc ids for the given port channels.
This method updates the vpcId with the given newVpcId and associate corresponding Vpc with the given SwitchedEtherChannelNetworkInterface InstanceNameId.
ValidationException is thrown if any of the following situation occurs:
- If portChannelInstanceNameId is null.
- If portChannelInstanceNameId is not of type SwitchedEtherChannelNetworkInterface InstanceNameId.
IntegrityException is thrown if the given portchannels do not exist in device.

Parameters
opContext—Operational context
portChannelInstanceNameId—InstanceNameId of SwitchedEtherChannelNetworkInterface
Chapter 39      VpcApp Service

modifyVpcs

Modifies any association changes on Vpc.
This method updates the server for any association changes in Vpc end points and Peer-link end points
ValidationException is thrown if any of the following situation occurs:

- If the vPCs is null or empty.
- If the vPCs contains one or more null element, or the collection contains objects that are not of type VpcEndPoint.

IntegrityException is thrown if any of the following situation occurs:

- If the given Vpc does not exist in device.

Parameters

opContext—Operational context
vpcs—List of Modified Vpc

Return Value

void

synchronizeGlobalSettingsForPrimaryAndSecondary

Synchronizes vPC related primary network element global configurations with secondary network element.
This method copies vPC related primary global configurations to secondary.
ValidationException is thrown if any of the following situation occurs:

- If primaryNeId is null.
- If primaryNeId is not of type AbstractNetworkElement InstanceNameId.
- If secondaryNeId is null.
- If secondaryNeId is not of type AbstractNetworkElement InstanceNameId.

IntegrityException is thrown if the given Vpc do not exist in device.

Parameters

opContext—Operational context
primaryNeId—InstanceNameId of AbstractNetworkElement
secondaryNeId—InstanceNameId of AbstractNetworkElement
synchronizePrimaryAndSecondary

Synchronizes vPC related primary network element configurations with secondary network element. This method copies vPC related primary configurations to secondary. ValidationException is thrown if any of the following situation occurs:

- If primaryNeId is null.
- If primaryNeId is not of type AbstractNetworkElement InstanceNameId.
- If vpcInstanceNameId is null.
- If vpcInstanceNameId is not of type VpcEndPoint InstanceNameId.

IntegrityException is thrown if the given Vpc do not exist in device.

Parameters

opContext—Operational context
primaryNeId—InstanceNameId of AbstractNetworkElement
vpcInstanceNameId—InstanceNameId of VpcEndPoint

Return Value

void
VrfApp Service

This chapter describes the DCNM web services’ API methods for the VrfApp service.

Information About VrfApp Service

The Layer 3 functionality in a VDC can be further virtualized into multiple routing domains using virtual routing and forwarding (VRF) instances. A separate routing and forwarding table is maintained for each VRF. A Layer 3 interface (logical or physical) in a VDC can belong to exactly one VRF. A VRF is local to a VDC and each VDC can contain multiple VRFs. By default, each VDC will contain a VRF. By default, all Layer 3 interfaces belong to the default VRF. In addition, for each VDC, a management VRF is created automatically.

Cisco NX-OS supports a VRF-lite implementation. VRF-lite enables a service provider to support two or more VPNs, where the IP addresses can overlap among the VPNs. VRF-lite uses input interfaces to distinguish routes for different VPNs and forms virtual packet-forwarding tables by associating one or more Layer 3 interfaces with each VRF. Interfaces in a VRF can be either physical, such as Ethernet ports, or logical, such as a VLAN SVI. A Layer 3 interface cannot belong to more than one VRF.

With VRF-lite, multiple customers can share one customer edge (CE), and only one physical link is used between the CE and the provider edge (PE). The shared CE maintains separate VRF tables for each customer and routes packets for each customer based on its own routing table. VRF-lite extends limited PE functionality to a CE device, giving it the ability to maintain separate VRF tables to extend the privacy and security of a VPN to the branch office.

addIpNetworkInterfaces

Add IpNetworkInterfaces to VRF.

Parameters

vrffInstanceNameId—InstanceId of VRF
ipNetworkInterfaceCol—List of IpNetworkInterface objects to be added.

Return Value

void
addIpNetworkInterfacesToDefaultVrf

Add IpNetworkInterfaces to Default VRF.

**Parameters**
ipNetworkInterfaceCol—List of IpNetworkInterface objects to be added.

**Return Value**
void

addIpv4StaticRoutes

Add a list of static routes to the specified network element.

**Parameters**
opContext—Operational context
InstanceNameId—Network element
vrfName—String specifying VRF name
Ipv4StaticRoute—List of static routes to add

**Return Value**
List of InstanceNameId

addIpv6StaticRoutes

Add a list of static routes to the specified network element.

**Parameters**
opContext—Operational context
InstanceNameId—Network element
vrfName—String specifying VRF name
Ipv6StaticRoute—List of static routes to add

**Return Value**
List of InstanceNameId

createVrfs

Create VRF on a VDC.

**Parameters**
neInstanceNameId—InstanceNameId of VDC
**deleteVrfs**

Removes VRFs from VDC.

**Parameters**

vrfInstanceNameIdCol—List of InstanceNameId of vrf to be deleted.

**Return Value**

void

**getAllNetworkElements**

Returns list VDCs in the network. This method will return neIds of VDCs that have the VRF created.

**Return Value**

List of VDC elements that have VRF enabled

**getIpNetworkInterfaces**

Returns list of IpNetworkInterfaces in a VRF.

**Parameters**

vrfInstanceNameId—Instance name ID of Vrf to be queried.

**Return Value**

List of IpNetworkInterface Objects.

**getMulticastRoutingInstancesForVrf**

Returns list of protocols instance Identifier that are enabled in a VRF.

**Parameters**

vrfInstanceNameId—Instance name ID of Vrf to be queried.

**Return Value**

List of RoutingInstances Objects for VRF.
getUnicastRoutingInstancesForVrf

Returns list of protocols instance Identifier that are enabled in a VRF.

Parameters
vrfInstanceNameId—Instance name ID of Vrf to be queried.

Return Value
List of RoutingInstances Objects for VRF.

getVrfsInNetworkElement

Returns list of VRFs in a VDC.

Parameters
neInstanceNameId—Instance name ID of network element for which Vrf are to be queried.

Return Value
List of Vrf.

removeStaticRoutes

Removes a list of static routes.

Parameters
opContext—Operational context
Ipv4StaticRoute—List of static routes to remove

Return Value
void