



Perl Client

This appendix provides information about the Perl client, installing ActiveState Perl, and the SOAP:Lite package.

This appendix contains the following sections:

- [Installing Perl](#)
- [Installing SOAP:Lite](#)
- [Running the Perl Clients](#)
- [Running the Zone Client](#)
- [Running the Statistics Client](#)

Before executing the Perl script, install ActiveState Perl and the SOAP:Lite package.

Installing Perl

Step 1 Download Perl from <http://www.perl.org/get.html>



Note Choose and download the appropriate Perl version for your operating system.

Step 2 Run the downloaded executable file to install Perl.

Step 3 To verify whether Perl has been installed properly, use the **perl -v** CLI command in the command prompt.
If the installation is proper, the details of the installed version of Perl are displayed.

Installing SOAP:Lite

Download SOAP:Lite from <http://search.cpan.org/dist/SOAP-Lite/>.



Note Choose and download the appropriate SOAP:Lite version for your operating system.

To install SOAP:lite, follow these steps:

Send documentation comments to dcnm-san-docfeedback@cisco.com

-
- Step 1** Using the command prompt, browse to the directory where the Makefile.PL is located in the downloaded SOAP:Lite package.
- Step 2** Use the `perl Makefile.PL -noprompt` CLI command to install the SOAP:Lite package.
- Step 3** Download nmake utility for Windows from <ftp://ftp.microsoft.com/Softlib/MSLFILES/Nmake15.exe>.
- Step 4** Extract the downloaded Nmake15.exe
You will find two files, NMAKE.exe and NMAKE.ERR.
Copy both the NMAKE.exe and NMAKE.ERR files to the installed Perl bin directory.
- Step 5** Complete the SOAP:Lite installation by entering the following commands for Windows and Linux machines:

```
nmake
nmake test
nmake install
```

For more information on installing SOAP:Lite, see <http://soaplite.com/install.html>.

Running the Perl Clients



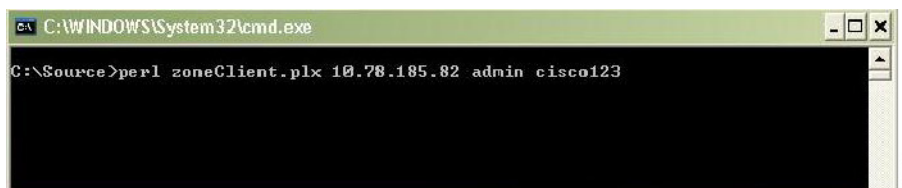
Note

To run a Perl client, the client file and its module file need to be in the same Perl folder.

From the Perl folder where the source files exist, enter the following command:

```
perl Client-File-Name FM-Server-IP Username password
```

Figure B-1 Running a Perl Client



Running the Zone Client

To run the zone client, the files `zoneClient.plx` and `zoneModule.plx` need to be in same directory.

From the Perl folder where the source files exist, enter the following command:

```
perl zoneClient.plx FM-Server-IP Username password
```

Running the Statistics Client

To run the statistics client, the files `statistics.plx` and `statisticsModule.plx` need to be in same directory.

From the Perl folder where the source files exist, enter the following command:

Send documentation comments to dcnm-san-docfeedback@cisco.com

```
perl statistics.plx FM-Server-IP Username password
```

Sample Code



Note

- The sample code provided here works only with DCNM release 6.1(x) and later. For prior DCNM versions, you need to make changes to the namespace base on the WSDL.
- To collect statistics related data, you need to turn on the PM.

Example B-1 ZoneClient.plx

```
# File: zoneClient.plx
# Name: Shailin Saraiya
# Description: zoneClient

# Variables

my $username;
my $password;
my $vsanId = 01;
my $vsan_wwn = '20:01:00:0d:ec:19:6a:81';
my $wwn = '20:00:00:0d:ec:19:6a:80';
my $zoneName = 'test5';
my $readOnly = 0;
my $broadcast = 0;
my $qos = 0;
my $qosPriority = -1;
my $zoneMemberType = 1;
my $zoneMemberFormat = 1;
my $zoneMemberIvrFabricIndex = -1;
my $zoneMemberIvrVsanIndex = -1;
my $zoneMemberId = 'bla2334455667788';
my $zoneLastModtime = 0;
my $zoneMemberLunId = '';
my $zoneMemberDbId = 0;
my $zoneisIvr = 0;
my $zoneSetName = 'ZoneSet5';
my $active = 0;

require 'zoneModule.plx';
($url,$username,$password) = @ARGV;
# Getting Token
my $token = getToken($url,$username,$password);
if(!$token) {
print "\nError in getToken call \n";
goto exit;
}
# Creating Zone
my $zoneIndex =
createZone($token,$vsanId,$vsan_wwn,$wwn,$zoneName,$readOnly,$broadcast,$qos,$qosPriority)
;

if(!$zoneIndex) {
print "\nError in createZone call \n";
goto exit;
}
# Adding Member to Zone
```

Send documentation comments to dcnm-san-docfeedback@cisco.com

```

my $status =
addZoneMemberToZone($token,$vsanId,$vsan_wwn,$wwn,$zoneName,$zoneMemberType,$zoneMemberFor
mat,$zoneMemberIvrFabricIndex,$zoneMemberIvrVsanIndex,$zoneMemberId,$zoneMemberLunId);
if(!$status) {
print "\nError in addZoneMemberToZone call \n";
goto exit;
}
# Creating ZoneSet
my $zoneSetIndex = createZoneSet($token,$vsanId,$vsan_wwn,$wwn,$zoneSetName);
if(!$zoneSetIndex) {
print "\nError in createZoneSet call \n";
goto exit;
}
# Add Zone to ZoneSet
my $status =
addZoneToZoneset($token,$vsanId,$vsan_wwn,$wwn,$zoneIndex,$zoneName,$readOnly,$broadcast,$
qos,$qosPriority,$zoneMemberDbId,$zoneMemberType,$zoneLastModtime,$zoneMemberLunId,$zoneis
Ivr,$zoneSetIndex,$zoneSetName,$active);
if(!$status) {
print "\nError in addZoneToZoneset call \n";
goto exit;
}
# Activate ZoneSet
$status = activateZoneset($token,$vsanId,$vsan_wwn,$wwn,$zoneSetName);
if(!$status) {
print "\nError in activateZoneset call \n";
goto exit;
}
else {
print "successful \n";
}
exit:

```

Example B-2 ZoneModule.plx

```

use SOAP::Lite;
# Get Token
sub getToken {
my $url;
my $username;
my $password;
($url,$username,$password) = @_;
# Setting uri and proxy

my $client = SOAP::Lite
#->uri('http://ep.jaxws.dcbu.cisco.com/')
->proxy('https://'. $url. '/LogonWSService/LogonWS?wsdl')
->ns('http://ep.jaxws.dcbu.cisco.com/', 'namesp1');
my $username = SOAP::Data->type('string')->name('username')->value($username);
my $password = SOAP::Data->type('string')->name('password')->value($password);
my $expiration = SOAP::Data->type('long')->name('expiration')->value(100000000);
# Calling requestToken method
my $result = $client->requestToken($username,$password,$expiration);
# check for error
unless ($result->fault) {
my $token = $result->result();
return $token;
} else {
# error handling
print join ' ',
$result->faultcode,

```

Send documentation comments to dcnm-san-docfeedback@cisco.com

```

        $result->faultstring,
        $result->faultdetail;
    return 0
    }
}

# Create Zone

sub createZone {

    my $token;
    my $arg_vsanId;
    my $arg_vsanwwn;
    my $arg_wwn;
    my $arg_zoneName;
    my $arg_readOnly;
    my $arg_broadcast;
    my $arg_qos;
    my $arg_qosPriority;

    ($token, $arg_vsanId, $arg_vsanwwn, $arg_wwn, $arg_zoneName, $arg_readOnly, $arg_broadcast, $arg_
    qos, $arg_qosPriority) = @_;
    my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
        SOAP::Data->name("vsanID" => $arg_vsanId),
        SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value" =>
        $arg_vsanwwn))->type("Wwn"))->type("VsanKey"));
    my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
        SOAP::Data->name("wwn" =>
        \SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn))->type("Wwn"))->type("WwnKey"));
    my $zName = SOAP::Data->name("zoneName"=>$arg_zoneName);
    my $readOnly = SOAP::Data->name("readOnly")->value($arg_readOnly);
    my $broadcast = SOAP::Data->name("broadcast")->value($arg_broadcast);
    my $qos = SOAP::Data->type('boolean')->name("qos")->value($arg_qos);
    my $qosPriority =
    SOAP::Data->type('int')->name("qosPriority")->value($arg_qosPriority);

    # Setting header
    my $header =
    SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
    om')->type('string');
    my $client = SOAP::Lite
        #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
        ->proxy('https://'. $url .'/ZoneManagerWSService/ZoneManagerWS?wsdl')
        ->ns('http://ep.san.jaxws.dcbu.cisco.com/', 'namesp2');
    # Calling createZone method
    my $result =
    $client->createZone($header, $VsanKey, $Wwn, $zName, $readOnly, $broadcast, $qos, $qosPriority);
    unless ($result->fault) {
        return $result->valueof('//index');
    } else {
        print join ', ',
            $result->faultcode,
            $result->faultstring,
            $result->faultdetail;
        return 0;
    }
}

# Adding Zone Member To Zone

sub addZoneMemberToZone {
    my $token;
    my $arg_vsanId;

```

Send documentation comments to dcnm-san-docfeedback@cisco.com

```

my $arg_wwn;
my $arg_vsanwwn;
my $arg_zoneName;
my $arg_zoneMemberType;
my $arg_zoneMemberFormat;
my $arg_zoneMemberIvrFabricIndex;
my $arg_zoneMemberIvrVsanIndex;
my $arg_zoneMemberId;
my $arg_zoneMemberLunId;

($token, $arg_vsanId, $arg_vsanwwn, $arg_wwn, $arg_zoneName, $arg_zoneMemberType, $arg_zoneMemberFormat, $arg_zoneMemberIvrFabricIndex, $arg_zoneMemberIvrVsanIndex, $arg_zoneMemberId, $arg_zoneMemberLunId) = @_;
my @hex = ($arg_zoneMemberId =~ /(..)/g);
my @dec = map { hex($_) } @hex;
my @zoneMemberId = map { pack('C', $_) } @dec;
my $arg_zoneMemberId = join("", @zoneMemberId);

my $client = SOAP::Lite
    #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
    ->proxy('https://'. $url. '/ZoneManagerWSService/ZoneManagerWS?wsdl')
    ->ns('http://ep.san.jaxws.dcbu.cisco.com/', 'namesp3');
my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
    SOAP::Data->name("vsanID" => $arg_vsanId),
    SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value" =>
$arg_vsanwwn))->type("Wwn"))->type("VsanKey"));
my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
    SOAP::Data->name("wwn" =>
\SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn))->type("Wwn"))->type("WwnKey"));
my $zName = SOAP::Data->name("zoneName"=>$arg_zoneName);
my $zMemberType = SOAP::Data->name("zoneMemberType")->value($arg_zoneMemberType);
my $zMemberFormat =
SOAP::Data->name("zoneMemberFormat")->value($arg_zoneMemberFormat);
my $zMemberIvrFabricIndex =
SOAP::Data->name('zoneMemberIvrFabricIndex')->value($arg_zoneMemberIvrFabricIndex);
my $zMemberIvrVsanIndex =
SOAP::Data->type('int')->name("zoneMemberIvrVsanIndex")->value($arg_zoneMemberIvrVsanIndex);
my $zMemberId = SOAP::Data->name("zoneMemberId")->value($arg_zoneMemberId);
my $zMemberLunId = SOAP::Data->name("zoneMemberLunId")->value($arg_zoneMemberLunId);

my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.com')->type('string');
my $result =
$client->addZoneMemberToZone($header, $VsanKey, $Wwn, $zName, $zMemberType, $zMemberFormat, $zMemberIvrFabricIndex, $zMemberIvrVsanIndex, $zMemberId, $zMemberLunId);
unless ($result->fault) {
    return 1;
}
else {
print join ' ',
    $result->faultcode,
    $result->faultstring,
    $result->faultdetail;
return 0;
}
}

# Creating ZoneSet

sub createZoneSet {

```

Send documentation comments to dcnm-san-docfeedback@cisco.com

```

my $token;
my $arg_vsanId;
my $arg_vsanwwn;
my $arg_wwn;
my $arg_zoneSetName;

($token,$arg_vsanId,$arg_vsanwwn,$arg_wwn,$arg_zoneSetName) = @_;
my $client = SOAP::Lite
    #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
    ->proxy('https://'. $url. '/ZoneManagerWSService/ZoneManagerWS?wsdl')
    ->ns('http://ep.san.jaxws.dcbu.cisco.com/', 'namesp4');
# Creating complex data type

my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
    SOAP::Data->name("vsanID" => $arg_vsanId),
    SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value" =>
$arg_vsanwwn)))->type("Wwn")))->type("VsanKey");
my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
    SOAP::Data->name("wwn" =>
\SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn)))->type("Wwn")))->type("WwnKey");
my $zSetName = SOAP::Data->name("zoneSetName"=>$arg_zoneSetName);

# Setting header

my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');

# Calling createZone method

my $result = $client->createZoneSet($header,$VsanKey,$Wwn,$zSetName);

unless ($result->fault) {
    return $result->valueof('//index')." \n";
} else {
    print join ', ',
        $result->faultcode,
        $result->faultstring,
        $result->faultdetail;
return 0;
}
}

# Adding Zone to ZoneSet
sub addZoneToZoneset {
    my $token;
    my $arg_vsanId;
    my $arg_vsanwwn;
    my $arg_wwn;
    my $arg_index;
    my $arg_zoneName;
    my $arg_readOnly;
    my $arg_broadcast;
    my $arg_qos;
    my $arg_qosPriority;
    my $arg_zoneMemberDbId;
    my $arg_zoneLastModtime;
    my $arg_memberType;
    my $arg_zoneMemberLunId;
    my $arg_isIvr;
    my $arg_zoneSetindex;
    my $arg_zoneSetName;

```

Send documentation comments to dcnm-san-docfeedback@cisco.com

```

my $active;

($token, $arg_vsanId, $arg_vsanwwn, $arg_wwn, $arg_index, $arg_zoneName, $arg_readOnly, $arg_broad-
dcast, $arg_qos, $arg_qosPriority, $arg_zoneMemberDbId, $arg_memberType, $arg_zoneLastModtime, $
arg_zoneMemberLunId, $arg_isIvr, $arg_zoneSetindex, $arg_zoneSetName, $active) = @_;

my $client = SOAP::Lite
    #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
    ->proxy('https://'. $url. '/ZoneManagerWSService/ZoneManagerWS?wsdl')
    ->ns('http://ep.san.jaxws.dcbu.cisco.com/', 'namesp5');
# Creating complex data type

my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
    SOAP::Data->name("vsanID" => $arg_vsanId),
    SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value" =>
$arg_vsanwwn)))->type("Wwn"))->type("VsanKey");
my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
    SOAP::Data->name("wwn" =>
\SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn)))->type("Wwn"))->type("WwnKey");
my $zoneDO = SOAP::Data->name("zoneDo"=> \SOAP::Data->value(
    SOAP::Data->name("index"=>$arg_index),
    SOAP::Data->name("name"=>$arg_zoneName),
    SOAP::Data->name("readonly"=>$arg_readonly),
    SOAP::Data->name("qos"=>$arg_qos),
    SOAP::Data->name("qosPriority"=>$arg_qosPriority),
    SOAP::Data->name("broadcast"=>$arg_broadcast),
    SOAP::Data->name("active"=>$active),
    SOAP::Data->name("zoneLastModtime"=>$arg_zoneLastModtime),
    SOAP::Data->name("zoneMemberDbId"=>$arg_zoneMemberDbId),
    SOAP::Data->name("vsanId"=>$arg_vsanId),
    SOAP::Data->name("isIvr"=>$arg_isIvr),
    SOAP::Data->name("memberType"=>$arg_memberType),
    SOAP::Data->name("lunId"=>$arg_zoneMemberLunId)))->type("zone");
my $zoneSetDO = SOAP::Data->name("zoneSetDo"=> \SOAP::Data->value(
    SOAP::Data->name("index"=>$arg_zoneSetindex),
    SOAP::Data->name("name"=>$arg_zoneSetName),
    SOAP::Data->name("active"=>$active),
    SOAP::Data->name("zoneLastModified"=>$arg_zoneLastModtime)))->type("ZoneSet");

# Setting header

my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');

# Calling addZonetoZoneSet method

my $result = $client->addZoneToZoneset($header, $VsanKey, $Wwn, $zoneDO, $zoneSetDO);

unless ($result->fault) {
    return 1;
} else {
    print join ', ',
        $result->faultcode,
        $result->faultstring,
        $result->faultdetail;
    return 0;
}
}

#Activating ZoneSet

sub activateZoneset {

```


Send documentation comments to dcnm-san-docfeedback@cisco.com

```

my $token;
my $arg_vsanId;
my $arg_vsanwwn;
my $arg_wwn;
my $arg_zoneSetName;

($token,$arg_vsanId,$arg_vsanwwn,$arg_wwn,$arg_zoneSetName) = @_;

my $client = SOAP::Lite
    #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
    ->proxy('https://'. $url. '/ZoneManagerWSService/ZoneManagerWS?wsdl')
    ->ns('http://ep.san.jaxws.dcbu.cisco.com/', 'namesp6');
# Creating complex data type

my $VsanKey = SOAP::Data->name("vsanKey" => \SOAP::Data->value(
    SOAP::Data->name("vsanID" => $arg_vsanId),
    SOAP::Data->name("pWwn" => \SOAP::Data->value(SOAP::Data->name("value"
=> $arg_vsanwwn)))->type("Wwn"))->type("VsanKey");
my $Wwn = SOAP::Data->name("switchKey" => \SOAP::Data->value(
    SOAP::Data->name("wwn" =>
\SOAP::Data->value(SOAP::Data->name("value"=>$arg_wwn)))->type("Wwn"))->type("WwnKey");
my $zSetName = SOAP::Data->name("zoneSetName"=>$arg_zoneSetName);

# Setting header

my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.c
om')->type('string');

# Calling activateZoneset method

my $result = $client->activateZoneset($header,$VsanKey,$Wwn,$zSetName);
unless ($result->fault) {
    return 1;
} else {
    print join ', ',
        $result->faultcode,
        $result->faultstring,
        $result->faultdetail;
    return 0;
}
}
1;

```

Example B-3 StatisticsClient.plx

```

# File: StatisticsClient.plx
# Name: Shailin Saraiya
# Description: Statistics Client

# Variables

my $username;
my $password;
my $url;
my $fabricDbId = -1;
my $switchDbId = -1;
my $vsanDbId = -1;
my $sortField = 'rxTxStr';
my $sortType = 'DESC';

```

Send documentation comments to dcnm-san-docfeedback@cisco.com

```

my $limit = NaN;
my $groupId = -1;
my $isGroup = 0;
my $networkType = 'SAN';
my $filterStr = '';
my $temp = '24 Hours';
my $startIdx = 0;
my $recordSize = 45;
my $interval = 1;
require 'statisticsModule.plx';

($url,$username,$password) = @ARGV;

# Getting Token
my $token = getToken($url,$username,$password);
($rrdFile,$fid,$pmType) =
getIs1StatList($token,$fabricDbId,$switchDbId,$vsanDbId,$sortField,$sortType,$limit,$groupId,
$isGroup,$networkType,$filterStr,$temp,$startIdx,$recordSize);
for (my $i=0;$i<1;$i++) {
    getPmChartData($token,@$rrdFile[$i],@$pmType[$i],@$fid[$i],$interval) . "\n";
}

```

Example B-4 StatisticsModule.plx

```

use SOAP::Lite;
use Time::Local;

# Get Token
$url = '';

sub getToken {
    my $username;
    my $password;
    ($url,$username,$password) = @_;

    # Setting uri and proxy

    my $client = SOAP::Lite
    #->uri('http://ep.jaxws.dcbu.cisco.com/')
    ->proxy('https://'. $url. '/LogonWSService/LogonWS?wsdl')
    -> ns('http://ep.jaxws.dcbu.cisco.com/', 'namespl');
    #-> autotype(0)
    #-> readable(1);
    my $username = SOAP::Data->type('string')->name('username')->value($username);
    my $password = SOAP::Data->type('string')->name('password')->value($password);
    my $expiration = SOAP::Data->type('long')->name('expiration')->value(100000000);

    # Calling requestToken method

    my $result = $client->requestToken($username,$password,$expiration);
    # check for error
    unless ($result->fault) {
        my $token = $result->result();
        $token;
    } else {

    # error handling
    print join ', ',
        $result->faultcode,
        $result->faultstring,

```

Send documentation comments to dcnm-san-docfeedback@cisco.com

```

        $result->faultdetail;
    }
}
sub getIslStatList {
    my $token;
    my $arg_fabricDbId;
    my $arg_switchDbId;
    my $arg_vsanDbId;
    my $arg_sortField;
    my $arg_sortType;
    my $arg_limit;
    my $arg_groupId;
    my $arg_isGroup;
    my $arg_networkType;
    my $arg_filterStr;
    my $arg_interval;
    my $arg_startIdx;
    my $arg_recordSize;

    ($token, $arg_fabricDbId, $arg_switchDbId, $arg_vsanDbId, $arg_sortField, $arg_sortType, $arg_limit, $arg_groupId, $arg_isGroup, $arg_networkType, $arg_filterStr, $arg_interval, $arg_startIdx, $arg_recordSize) = @_;
    my $DbFilter= SOAP::Data->name("arg0" => \SOAP::Data->value(
        SOAP::Data->name("fabricDbId" => $arg_fabricDbId),
        SOAP::Data->name("switchDbId" => $arg_switchDbId),
        SOAP::Data->name("vsanDbId" => $arg_vsanDbId),
        SOAP::Data->name("sortField" => $arg_sortField),
        SOAP::Data->name("sortType" => $arg_sortType),
        SOAP::Data->name("limit" => $arg_limit),
        SOAP::Data->name("groupId" => $arg_groupId),
        SOAP::Data->name("isGroup" => $arg_isGroup)->type("boolean"),
        SOAP::Data->name("networkType" => $arg_networkType),
        SOAP::Data->name("filterStr" => $arg_filterStr))->type("DbFilter");

    my $interval = SOAP::Data->type("string")->name("arg1")->value($arg_interval);
    my $data_startIdx = SOAP::Data->name("arg2"=>$arg_startIdx);
    my $data_recordSize = SOAP::Data->name("arg3"=>$arg_recordSize);

    # Setting header

    my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.com')->type('string');
    #my $header = SOAP::Header->name('token')->prefix('m')->value($token)->type('string');
    my $client = SOAP::Lite
        #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
        ->proxy('https://'. $url. '/StatisticsWSService/StatisticsWS?wsdl')
            -> ns('http://ep.san.jaxws.dcbu.cisco.com/', 'namesp2');
        #-> autotype(0)
        #-> readable(1);

    # Calling getIslStatDataLength method
    my $result =
$client->getIslStatList($header, $DbFilter, $interval, $data_startIdx, $data_recordSize);
    unless ($result->fault) {
        print "\ngetIslStatList\n\n";
        my @fabric = $result->valueof('//item/fabric');
        my @title = $result->valueof('//item/title');
        my @maxTxStr = $result->valueof('//item/maxTxStr');
        my @maxRxStr = $result->valueof('//item/rxTxStr');
        my @rxTx = $result->valueof('//item/rxTx');
    }
}

```

Send documentation comments to dcnm-san-docfeedback@cisco.com

```

my $count = @fabric;

for(my $i=0;$i<10;$i++)
{
    print "-----\n";
    print "Fabric -> " . $fabric[$i] . "\n";
    print "Title -> " . $title[$i] . "\n";
    print "maxTxStr -> " . $maxTxStr[$i] . "\n";
    print "maxRxStr -> " . $maxRxStr[$i] . "\n";
    print "rxTx -> " . $rxTx[$i] . "\n";
    print "-----\n";
}

my @rrdFile = $result->valueof('//item/rrdFile');
my @fid = $result->valueof('//item/fid');
my @pmType = $result->valueof('//item/pmtype');
return (\@rrdFile,\@fid,\@pmType);

} else {
    print join ', ',
        $result->faultcode,
        $result->faultstring,
        $result->faultdetail;
}

}

sub getPmChartData {
    my $token;
    my $arg_rrd;
    my $arg_pmType;
    my $arg_fid;
    my $arg_interval;
    ($token,$arg_rrd,$arg_pmType,$arg_fid,$arg_interval) = @_;
    my $data_rrd = SOAP::Data->type("string")->name("arg0")->value($arg_rrd);
    my $data_pmType = SOAP::Data->name("arg1")->value($arg_pmType);

    my $data_fid = SOAP::Data->name("arg2"=>$arg_fid);
    my $data_interval = SOAP::Data->name("arg3"=>$arg_interval);

    # Setting header
    my $header =
SOAP::Header->name('token')->prefix('m')->value($token)->uri('http://ep.cisco.dcbu.cisco.com')->type('string');
    my $client = SOAP::Lite
        #->uri('http://ep.san.jaxws.dcbu.cisco.com/')
        ->proxy('https://'. $url .'/StatisticsWSService/StatisticsWS?wsdl')
            -> ns('http://ep.san.jaxws.dcbu.cisco.com/', 'namesp3');
        #-> autotype(0)
        #-> readable(1);

    # Calling getPmChartData method
    my $result =
$client->getPmChartData($header,$data_rrd,$data_pmType,$data_fid,$data_interval);
    unless ($result->fault) {
        print "\n\nGetPmChartData\n\n";

        my @item = $result->valueof('//item/item');
        for(my $i=2;$i<30;$i=$i+3) {
            print "-----\n";

```

Send documentation comments to dcnm-san-docfeedback@cisco.com

```
print "Rx -> ".$item[$i-2]."\n";
print "Tx -> ".$item[$i-1]."\n";
print "Time -> ".scalar localtime($item[$i]) . "\n";
print "-----\n";
}
} else {
print join ', ',
    $result->faultcode,
    $result->faultstring,
    $result->faultdetail;
}
}
1;
```

Send documentation comments to dcnm-san-docfeedback@cisco.com