NSO Resource Manager

4.2.5

API Guide

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices**.

Contents

Preface

Abstract

This Cisco Network Service Orchestrator – Resource Manager – Application Programmable Interface documents describes APIs exposed by RM package, that you can use to allocate IPs from IP resource pool and to allocate the IDs from ID resource pools respectively.

Audience

This document is intended for Cisco Advanced Services developers, network engineers, and system engineers to install the RM package inside NSO and then utilize the APIs exposed by RM package to allocate and manage IP subnets and ID as required by other CFPs installed alongside this RM package inside the NSO.

Additional Documentation

This documentation requires the reader to have a good understanding of NSO and its usage as described in the NSO documentation.

Sl. No.	Documentation
1.	Cisco NSO Installation Guide
2.	Cisco NSO User Guide

Bias-free Documentation Policy

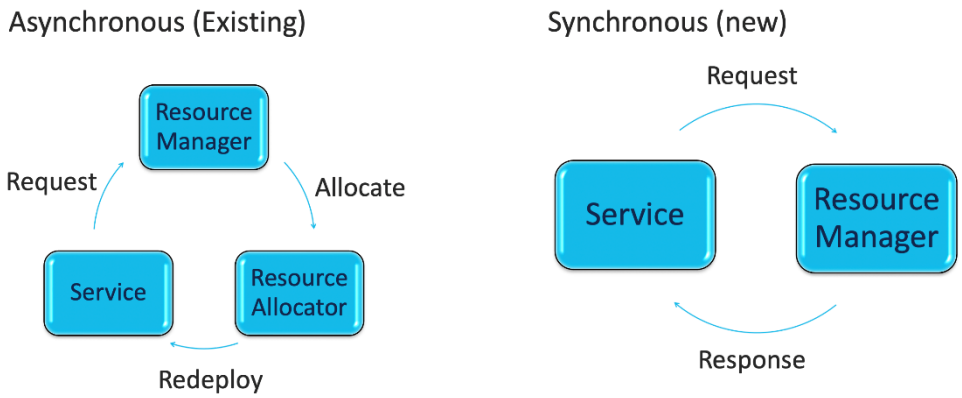
Cisco follows a bias-free documentation policy. According to this policy, Cisco treats all persons with respect—regardless of race, color, ancestry, national origin, age, sex, citizenship, veteran status, marital status, sexual orientation, physical or mental ability, religious creed, or medical condition. Language or graphic elements that offend others violate our business philosophy and our company policy.

Resource Manager IP/ID Allocation APIs

The APIs exposed by Resource Manager package are used to allocate IP subnets and IDs from the IP and ID resource pools respectively by the applications requesting the resources. The APIs help to allocate, update, or deallocate the resources. You can make API calls to the resource pools as long as the pool is not exhausted of the resources. If the pool is exhausted of the resources or if the referenced pool does not exist in the database when there is a request, the allocation raises an exception.

When a service makes multiple resource allocations from a single pool, the optional 'name' parameter allows the service to distinguish the different allocations. By default, the parameter value is an empty string.

Resource allocation can be synchronous or asynchronous.



The synchronized allocation API request uses a Reactive-Fast-Map to allocate resources and still manages the interface to look synchronous. This means that as you create any allocation request from Northbound, you can see the allocation results, such as requested IP subnet/ID in the same transaction. If a NB is making an allocation request, and in the same transaction a configuration is being applied to a specific device, the commit dry run receives the request response, and the response is processed by the RM and the configurations are pushed to the device in the same transaction. Thus, the NB user can see the get modification in the commit dry run.

During a resource request, the resource is allocated and stored in the create callback. This allocation is visible to other services that are run in the same or subsequent transactions, and therefore avoids recreation of resource when the service is redeployed. Synchronous allocation does not require service re-deploy to read allocation. The same transaction can read allocation. Commit dry-run or get-modification displays the allocation details as output.

Example

The following is an example for Northbound service callback passed with required API parameters for both synchronous and asynchronous IPv4 allocations. The example uses **pool-example** package as reference. The request describes the details it uses, such as the pool, device. Each allocation has an allocation ID. In the following example, the allocating service pulls one IPv4 address from the IPv4 resource pool. The requesting service then uses this allocated IP address to set the interface address on the device southbound to NSO.

```
-----------------------------------------------
# NORTHBOUND SERVICE CALLBACK EXAMPLE - SYNC
# -----------------------------------------------
class AllocateCallbacks(Service):

    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('AllocateCallbacks create(service=', service._path, ')')
        self.log.info('requested allocation {} from {}'.format(service.device,
service.pool))
        service_xpath = ("/allocating-service-async:allocating-service-
async[name='{}']")
        propslist = ip_allocator.net_request(service,
                                service_xpath.format(service.name),
                                "admin",
                                service.pool,
                                service.ipv4,
                                service.subnet_size, False, "default", True,
proplist, self.log)

        # Check
        net = ip_allocator.net_read("admin", root, service.pool, service.ipv4,
service.name)
        self.log.info('Check n/w create(IP=', net, ')')

        if net:
            self.log.info('received device {} ip-address value from {} is
ready'.format(service.device, service.pool))
            template = ncs.template.Template(service)
            vars = ncs.template.Variables()
            vars.add("SERVICE", str(service.ipv4))
```

```
            vars.add("DEVICE_NAME", str(service.device))
            vars.add("IP", str(net))
            template.apply('device', vars)
        return propslist


# ------------------------------------------------
# NORTHBOUND SERVICE CALLBACK EXAMPLE - ASYNC
# ------------------------------------------------
class AllocateCallbacksAsync(Service):

    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        self.log.info('AllocateCallbacksAsync create(service=', service._path,
')')
        self.log.info('requested allocation {} from {}'.format(service.device,
service.pool))
        service_xpath = ("/allocating-service-async:allocating-service-
async[name='{}']")
        ip_allocator.net_request(service,
                                 service_xpath.format(service.name),
                                 tctx.username,
                                 service.pool,
                                 service.ipv4,
                                 service.subnet_size)
        # Check
        net=ip_allocator.net_read(tctx.username, root, service.pool,
service.ipv4)
        self.log.info('Check n/w create(IP=', net, ')')

        if net:
            self.log.info('received device {} ip-address value from {} is
ready'.format(service.device, service.pool))
            template = ncs.template.Template(service)
            vars = ncs.template.Variables()
            vars.add("SERVICE", str(service.ipv4))
            vars.add("DEVICE_NAME", str(service.device))
            vars.add("IP", str(net))
            template.apply('device', vars
```

The payloads below demonstrate Northbound service allocation request using Resource Manager synchronous and asynchronous flows. The API pulls one IP address from the IPv4 resource pool and sets the returned IP address on the interface IOS1 device. The payloads demonstrate both synchronous and asynchronous flows.

Synchronous flow

```
admin@ncs% load merge alloc-sync.xml
[ok]
admin@ncs% commit dry-run
cli {
    local-node {
        data  devices {
                  device ios1 {
                      config {
                          ip {
                              prefix-list {
             +                    prefixes sample {
             +                        permit 11.1.0.0/32;
             +                    }
             +                    prefixes sample1 {
             +                        permit 11.1.0.1/32;
             +                    }
             +                    prefixes sample3 {
             +                        permit 11.1.0.2/32;
             +                    }
             +                    prefixes sample4 {
             +                        permit 11.1.0.3/32;
             +                    }
                              }
                          }
                      }
                  }
              }
             +allocating-service sync-test-1 {
             +    device ios1;
             +    pool IPv4;
             +    subnet-size 32;
             +    ipv4 sample;
             +}
             +allocating-service sync-test-2 {
             +    device ios1;
             +    pool IPv4;
             +    subnet-size 32;
             +    ipv4 sample1;
             +}
             +allocating-service sync-test-3 {
             +    device ios1;
             +    pool IPv4;
             +    subnet-size 32;
             +    ipv4 sample3;
             +}
             +allocating-service sync-test-4 {
             +    device ios1;
             +    pool IPv4;
             +    subnet-size 32;
             +    ipv4 sample4;
             +}
```

```
    }
}
```

Asynchronous flow

```
admin@ncs% load merge alloc-async.xml
[ok]


admin@ncs% commit dry-run
cli {
    local-node {
        data  resource-pools {
              +    ip-address-pool IPv4 {
              +        allocation sample {
              +            username admin;
              +            allocating-service /allocating-service-
async[name='async-test'];
              +            redeploy-type default;
              +            request {
              +                subnet-size 32;
              +            }
              +        }
              +    }
               }
              +allocating-service-async async-test {
              +    device ios1;
              +    pool IPv4;
              +    subnet-size 32;
              +    ipv4 sample;
              +}
    }
}
```

IPv4 and IPv6 have separate IP pool types; there is no mixed IP pool. You can specify a **prefixlen** parameter for IP pools to allocate a net of a given size. The default value is the maximum prefix length of 32 and 128 for IPv4 and IPv6 respectively.

The following APIs are used in IPv4 and IPv6 allocations.

IP Allocations

Resource Manager exposes the API calls to request IPv4 and IPv6 subnet allocations from resource pool. These requests can be synchronous or asynchronous. This topic discusses the APIs for these flows.

The NSO Resource Manager interface and the resource allocator provide a generic resource allocation mechanism that works well with services. Each pool has an allocation list where services are expected to create instances to signal that they request an allocation. Request parameters are stored in the request container and the allocation response is written in the response container.

The APIs exposed by RM are implemented in Python as well as Java so the NB user can configure the service to be Java package or Python package and call the allocator API as

per the implementation. The NB user can also use NSO CLI to make allocation request to IP allocator RM package.

Using Java APIs for IP Allocations

This section covers the Java APIs exposed by RM package to the NB user to make IP subnet allocation requests.

Creating Asynchronous IP Subnet Allocation Requests

The asynchronous subnet allocation requests can be created for a requesting service with:

- The redeploy type set to default type or set to redeployType
- The CIDR mask length can be set to invert the subnet mask length for Boolean operations with IP addresses or set to not be able to invert the subnet mask length
- Pass the starting IP address of the subnet to the requesting service redeploy type (default/redeployType)

The following are the Java APIs for asynchronous IP allocation requests.

Default Asynchronous Request
Description

The requesting service **redeploy type** is **default** and CIDR mask length cannot be inverted for the subnet allocation request. Make sure the **NavuNode** service is the same node you get in service create. This ensures the back pointers are updated correctly and RFM works as intended.

<table>
<tr><td rowspan="6">void
com.tailf.pkg.ipaddressallocator.IPAddressAllocator.
subnetRequest(NavuNode service,
String poolName,
String username,
int cidrmask,
String id)</td><td colspan="3">API Parameters</td></tr>
<tr><td>Parameter</td><td>Type</td><td>Description</td></tr>
<tr><td>Service</td><td>NavuNode</td><td>NavuNode referencing the requesting service node</td></tr>
<tr><td>poolName</td><td>String</td><td>Name of the resource pool to request the subnet IP address from</td></tr>
<tr><td>Username</td><td>String</td><td>name of the user to use when redeploying the requesting service</td></tr>
<tr><td>cidrmask</td><td>Int</td><td>CIDR mask length of the requested subnet</td></tr>
<tr><td></td><td>id</td><td>String</td><td>Unique allocation ID</td></tr>
</table>

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(service, poolName, userName, cidrMask,
id);
```

Asynchronous Request with Invert CIDR Flag
Description

The requesting service redeploy type is default and CIDR mask length can be inverted for the subnet allocation request. Make sure the NavuNode service is the same node you get in service create. This ensures the back pointers are updated correctly and RFM works as intended.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(NavuNode service, String poolName, String username, int cidrmask, String id, boolean invertCidr)	**API Parameters**		
	Parameter	**Type**	**Description**
	Service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	Username	string	name of the user to use when redeploying the requesting service
	cidrmask	Int	CIDR mask length of requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	If boolean value is true, the subnet mask length is inverted.

Common Example for the usage of subnetRequest from service.

The below code example shows that subnetRequest method can be called from the service by different types parameter values getting from the service object.

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(service, poolName, userName, cidrMask,
id, invertCidr.booleanValue());
```

```
@ServiceCallback(servicePoint="ipaddress-allocator-test-servicepoint",
        callType=ServiceCBType.CREATE)
    public Properties create(ServiceContext context,
                             NavuNode service,
                             NavuNode ncsRoot,
                             Properties opaque)
                             throws DpCallbackException {
```

```
        LOGGER.info("IPAddressAllocatorTest Servicepoint is triggered");

        try {
            String servicePath = service.getKeyPath();
            CdbSession sess = cdb.startSession(CdbDBType.CDB_OPERATIONAL);
            try {
                String dPath = servicePath + "/deploys";
                int deploys = 1;

                if (sess.exists(dPath)) {
                    deploys = (int)((ConfUInt32)sess.getElem(dPath)).longValue();
                }
                if (sess.exists(servicePath)) { // Will not exist the first time
                    sess.setElem(new ConfUInt32(deploys + 1), dPath);
                }

                NavuLeaf size = service.leaf("subnet-size");
                if (!size.exists()) {
                    return opaque;
                }
                int subnetSize = (int)((ConfUInt8)service.leaf("subnet-
size").value()).longValue();

                String redeployOption = null;
                if (sess.exists(servicePath + "/redeploy-option")) {
                    redeployOption = ConfValue.getStringByValue(servicePath +
"/redeploy-option",
                                            service.leaf("redeploy-option").value());
                }

                System.out.println("IPAddressAllocatorTest redeployOption:" +
redeployOption);

                if (redeployOption == null) {
                    IPAddressAllocator.subnetRequest(service, "mypool",
                    "admin", subnetSize, "test");
                } else {
                    RedeployType redeployType = RedeployType.from(redeployOption);
                    System.out.println("IPAddressAllocatorTest redeployType:" +
redeployType);
                    IPAddressAllocator.subnetRequest(service, redeployType,
                    "mypool", "admin", subnetSize, "test", false);
                }

                boolean error = false;

                boolean allocated = sess.exists(servicePath + "/allocated");
                boolean ready =
                    IPAddressAllocator.responseReady(service.context(), cdb,
"mypool", "test");

                if (ready) {
                    try {
                        IPAddressAllocator.fromRead(cdb, "mypool", "test");
                    } catch (ResourceErrorException e) {
                        LOGGER.info("The allocation has failed");
                        error = true;
                    }
                }
```

```
                if (ready && !error) {
                    if (!allocated) {
                        sess.create(servicePath + "/allocated");
                    }
                } else {
                    if (allocated) {
                        sess.delete(servicePath + "/allocated");
                    }
                }
            } finally {
                sess.endSession();
            }
        } catch (Exception e) {
            throw new DpCallbackException("Cannot create service", e);
        }
        return opaque;
    }
}
```

Asynchronous Request with Invert CIDR Flag and Redeploy Type Description

The requesting service redeploy type is redeployType and CIDR mask length can be inverted for the subnet allocation request. Make sure the NavuNode service is the same node you get in service create. This ensures the back pointers are updated correctly and RFM works as intended.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(NavuNode service, RedeployType redeployType, String poolName, String username, int cidrmask, String id, boolean invertCidr)	**API Parameters**		
	Parameter	**Type**	**Description**
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	string	name of the user to use when redeploying the requesting service
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	If boolean value is true, the subnet mask length is inverted.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(service, redeployType, poolName,
userName, cidrMask, id, invertCidr.booleanValue());
```

Asynchronous Request with specific Start IP address
Description

Pass a startIP value to the default type of the requesting service redeploy. The subnet IP address begins with the provided IP address. Make sure that the NavuNode service is the same node you get in service create. This ensures that the back pointers are updated correctly and that the RFM works as intended.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator.
subnetRequest(NavuNode service,
String poolName,
String username,
String startIp,
int cidrmask,
String id,
boolean invertCidr)

API Parameters

Parameter	Type	Description
service	NavuNode	NavuNode referencing the requesting service node
poolName	String	Name of the resource pool to request the subnet IP address from
username	String	name of the user to use when redeploying the requesting service
startIP	String	Starting IP address for the requested subnet
cidrmask	Int	CIDR mask length of the requested subnet
id	String	Unique allocation ID
invertCidr	Boolean	If boolean value is true, the subnet mask length is inverted.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(service, poolName, userName, startIp,
cidrMask, id, invertCidr.booleanValue());
```

Asynchronous Request with specific Start IP address and Re-deploy Type Description

Pass a startIP value to the redeployType of the requesting service redeploy. The subnet IP address begins with the provided IP address. Make sure that the NavuNode service is the same node you get in service create. This ensures that the back pointers are updated correctly and that the RFM works as intended.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(NavuNode service, RedeployType redeployType, String poolName, String username, String startIp, int cidrmask, String id, boolean invertCidr)	API Parameters		
	Parameter	Type	Description
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	string	name of the user to use when redeploying the requesting service
	startIP	String	Starting IP address for the requested subnet
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	If boolean value is true, the subnet mask length is inverted.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(service, redeployType, poolName,
userName, startIp, cidrMask, id, invertCidr.booleanValue());
```

Asynchronous Request with Service Context Description

Create an asynchronous IP subnet allocation request with requesting service redeploy type as default and CIDR mask length cannot be inverted for the subnet allocation request. Make sure to use the service context you get in the service create callback. This method takes any NavuNode, should you need it.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(ServiceContext context, NavuNode service,	**API Parameters**		
	Parameter	**Type**	**Description**
	service	NavuNode	NavuNode referencing the requesting service node

String poolName, String username, int cidrmask, String id)	poolName	String	Name of the resource pool to request the subnet IP address from
	username	string	name of the user to use when redeploying the requesting service
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(context, service, poolName, userName,
cidrMask, id);
```

Asynchronous Request with Context and Re-deploy Type Description

Create an asynchronous IP subnet allocation request with requesting service redeploy type as redeployType and CIDR mask length can be inverted for the subnet allocation request. Make sure to use the service context you get in the service create callback. This method takes any NavuNode, should you need it.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(ServiceContext context, NavuNode service, RedeployType redeployType, String poolName, String username, int cidrmask, String id, boolean invertCidr)	**API Parameters**		
	Parameter	**Type**	**Description**
	Context	ServiceContext	ServiceContext referencing the requesting context the service was invoked in.
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	String	name of the user to use when redeploying the requesting service
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	If boolean value is true, the subnet mask length is inverted.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(context, service, redeployType,
poolName, userName, cidrMask, id, invertCidr.booleanValue());
```

Asynchronous Request with context and specific Start IP
Description

Pass a startIP value to the requesting service redeploy type, default. The subnet IP address begins with the provided IP address. CIDR mask length cannot be inverted for the subnet allocation request. Make sure to use the service context you get in the service create callback.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(ServiceContext context, NavuNode service, String poolName, String username, String startIp, int cidrmask, String id, boolean invertCidr)	API Parameters		
	Parameter	**Type**	**Description**
	Context	ServiceContext	ServiceContext referencing the requesting context the service was invoked in.
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	String	name of the user to use when redeploying the requesting service
	startIP	String	Starting IP address for the requested subnet
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	If boolean value is true, the subnet mask length is inverted.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(context, service, poolName, userName,
startIp, cidrMask, id, invertCidr.booleanValue());
```

Asynchronous Request with specific Start IP, context, Invert CIDR and Re-deploy Type Description

Pass a startIP value to the requesting service redeploy type, redeployType. The subnet IP address begins with the provided IP address. CIDR mask length can be inverted for the subnet allocation request. Make sure to use the service context you get in the service create callback.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(ServiceContext context, NavuNode service, RedeployType redeployType, String poolName, String username, String startIp, int cidrmask, String id, boolean invertCidr)	API Parameters		
	Parameter	**Type**	**Description**
	Context	ServiceContext	ServiceContext referencing the requesting context the service was invoked in.
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	string	name of the user to use when redeploying the requesting service
	startIP	String	Starting IP address for the requested subnet
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	If boolean value is true, the subnet mask length is inverted.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(context, service, redeployType,
poolName, userName, startIp, cidrMask, id, invertCidr.booleanValue());
```

Common exceptions raised by Java APIs for allocation not successful.

The API throws the following exception error if the requested resource pool does not exist.

ResourceErrorException

The API throws the following exception error if the requested resource pool is exhausted.

AddressPoolException

The API throws the following exception error if the requested netmask is invalid.

InvalidNetmaskException

Creating Synchronous or Asynchronous IP Subnet Allocation Requests

The **sync** parameter in the API determines if the allocation request is for a synchronous or asynchronous mode. Set the **sync** parameter to true for synchronous flow.

The subnet allocation requests can be created for a requesting service with:

- The redeploy type set to default type or redeployType type
- The CIDR mask length can be set to invert the subnet mask length for Boolean operations with IP addresses or set to not be able to invert the subnet mask length
- Pass the starting IP address of the subnet to the requesting service redeploy type (default/redeploytype)

The following are the Java APIs for synchronous or asynchronous IP allocation requests.

Default Java API for IP subnet allocation request
Description

The requesting service redeploy type is default and CIDR mask length can be inverted for the subnet allocation request. Set sync to true to make a synchronous allocation request with commit dry-run support. Make sure the NavuNode service is the same node you get in service create. This ensures the back pointers are updated correctly and RFM works as intended.

<table>
<tr><td rowspan="8">void
com.tailf.pkg.ipaddressallocator.IPAddressAllocator.
subnetRequest(NavuNode service,
String poolName,
String username,
int cidrmask,
String id,
boolean invertCidr,
boolean sync)</td><td colspan="3">API Parameters</td></tr>
<tr><td>Parameter</td><td>Type</td><td>Description</td></tr>
<tr><td>service</td><td>NavuNode</td><td>NavuNode referencing the requesting service node</td></tr>
<tr><td>poolName</td><td>String</td><td>Name of the resource pool to request the subnet IP address from</td></tr>
<tr><td>username</td><td>string</td><td>name of the user to use when redeploying the requesting service</td></tr>
<tr><td>cidrmask</td><td>Int</td><td>CIDR mask length of the requested subnet</td></tr>
<tr><td>id</td><td>String</td><td>Unique allocation ID</td></tr>
<tr><td>invertCidr</td><td>Boolean</td><td>Set value to true to invert the subnet mask length.</td></tr>
</table>

	sync	boolean	Set value to true to make a synchronous allocation request.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(service, poolName, userName, cidrMask,
id, invertCidr.booleanValue(), testSync.booleanValue());
```

Java API for IP subnet allocation request with redeploy type

Description

The requesting service redeploy type is redeployType and CIDR mask length can be inverted for the subnet allocation request. Set sync to true to make a synchronous allocation request with commit dry-run support. Make sure the NavuNode service is the same node you get in service create. This ensures the back pointers are updated correctly and RFM works as intended.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(NavuNode service, RedeployType redeployType, String poolName, String username, int cidrmask, String id, boolean invertCidr, boolean sync)	**API Parameters**		
	Parameter	**Type**	**Description**
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	String	name of the user to use when redeploying the requesting service
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	Set value to true to invert the subnet mask length.
	sync	boolean	Set value to true to make a synchronous allocation request.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(service, redeployType, poolName,
userName, cidrMask, id, invertCidr.booleanValue(),
testSync.booleanValue());
```

Java API for IP subnet allocation request with start IP address

Description

Pass a startIP value to the requesting service redeploy type, default. The subnet IP address begins with the provided IP address. Set sync to true to make a synchronous allocation request with commit dry-run support. Make sure that the NavuNode service is the same node you get in service create. This ensures that the back pointers are updated correctly and that the RFM works as intended.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(NavuNode service, String poolName, String username, String startIp, int cidrmask, String id, boolean invertCidr, boolean sync)	**API Parameters**		
	Parameter	**Type**	**Description**
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	string	name of the user to use when redeploying the requesting service
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	Set value to true to invert the subnet mask length.
	sync	Boolean	Set value to true to make a synchronous allocation request.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(service, poolName, userName, startIp,
cidrMask, id, invertCidr.booleanValue(), testSync.booleanValue());
```

Java API for IP subnet allocation request with redeploy type, start IP address and CIDR mask

Description

Pass a startIP value to the redeployType of the requesting service redeploy. The subnet IP address begins with the provided IP address. Set sync to true to make a synchronous allocation request with commit dry-run support. Make sure that the NavuNode service is the same node you get in service create. This ensures that the back pointers are updated correctly and that the RFM works as intended.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(NavuNode service, RedeployType redeployType, String poolName, String username, String startIp, int cidrmask, String id, boolean invertCidr, boolean sync)	**API Parameters**		
	Parameter	**Type**	**Description**
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	String	name of the user to use when redeploying the requesting service
	startIP	String	Starting IP address for the subnet allocation request
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	Set value to true to invert the subnet mask length.
	sync	Boolean	Set value to true to make a synchronous allocation request.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(service, redeployType, poolName,
userName, startIp, cidrMask, id, invertCidr.booleanValue(),
testSync.booleanValue());
```

Java API for IP subnet allocation request with service context

Description

Create an IP subnet allocation request with requesting service redeploy type as default and CIDR mask length cannot be inverted for the subnet allocation request. Make sure to use the

service context you get in the service create callback. Set sync to true to make a synchronous allocation request with commit dry-run support.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(ServiceContext context, NavuNode service, String poolName, String username, int cidrmask, String id, boolean invertCidr, boolean sync)	API Parameters		
	Parameter	**Type**	**Description**
	Context	ServiceContext	ServiceContext referencing the requesting context the service was invoked in.
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	String	name of the user to use when redeploying the requesting service
	startIP	String	Starting IP address for the subnet allocation request
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	Set value to true to invert the subnet mask length.
	sync	Boolean	Set value to true to make a synchronous allocation request.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(context, service, poolName, userName,
cidrMask, id, invertCidr.booleanValue(), testSync.booleanValue());
```

Java API for IP subnet allocation request with service context and redeploy type Description

Create an IP subnet allocation request with requesting service redeploy type as redeployType and CIDR mask length can be inverted for the subnet allocation request. Set sync to true to make a synchronous allocation request with commit dry-run support. Make sure to use the service context you get in the service create callback.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(ServiceContext context, NavuNode service, RedeployType redeployType, String poolName, String username, int cidrmask, String id, boolean invertCidr, boolean sync)	API Parameters		
	Parameter	**Type**	**Description**
	Context	ServiceContext	ServiceContext referencing the requesting context the service was invoked in.
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from
	username	String	name of the user to use when redeploying the requesting service
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	Set value to true to invert the subnet mask length.
	sync	Boolean	Set value to true to make a synchronous allocation request.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(context, service, redeployType,
poolName, userName, cidrMask, id, invertCidr.booleanValue(),
testSync.booleanValue());
```

Java API for IP subnet allocation request with service context and start IP address Description

Pass a startIP value to the requesting service redeploy type, default. The subnet IP address begins with the provided IP address. CIDR mask length can be inverted for the subnet allocation request. Set sync to true to make a synchronous allocation request with commit dry-run support. Make sure to use the service context you get in the service create callback.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(ServiceContext context, NavuNode service, String poolName, String username, String startIp, int cidrmask,	API Parameters		
	Parameter	**Type**	**Description**
	Context	ServiceContext	ServiceContext referencing the requesting context the service was invoked in.
	service	NavuNode	NavuNode referencing the requesting service node

String id, boolean invertCidr, boolean sync)	poolName	String	Name of the resource pool to request the subnet IP address from
	username	String	name of the user to use when redeploying the requesting service
	startIP	String	Starting IP address of the IP subnet allocation request
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	Set value to true to invert the subnet mask length.
	sync	Boolean	Set value to true to make a synchronous allocation request.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(context, service, poolName, userName,
startIp, cidrMask, id, invertCidr.booleanValue(),
testSync.booleanValue());
```

Java API for IP subnet allocation request with service context and start IP address and redeploy-type

Description

Pass a startIP value to the requesting service redeploy type, redeployType. The subnet IP address begins with the provided IP address. CIDR mask length can be inverted for the subnet allocation request. Set sync to true to make a synchronous allocation request with commit dry-run support. Make sure to use the service context you get in the service create callback.

void com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRequest(ServiceContext context, NavuNode service, RedeployType redeployType, String poolName, String username, String startIp, int cidrmask, String id,	API Parameters		
	Parameter	Type	Description
	Context	ServiceContext	ServiceContext referencing the requesting context the service was invoked in.
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the subnet IP address from

boolean invertCidr, boolean sync)	username	String	name of the user to use when redeploying the requesting service
	startIP	String	Starting IP address of the IP subnet allocation request
	cidrmask	Int	CIDR mask length of the requested subnet
	id	String	Unique allocation ID
	invertCidr	Boolean	Set value to true to invert the subnet mask length.
	sync	Boolean	Set value to true to make a synchronous allocation request.

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

IPAddressAllocator.subnetRequest(context, service, redeployType,
poolName, userName, startIp, cidrMask, id, invertCidr.booleanValue(),
testSync.booleanValue());
```

Common exceptions raised by Java APIs for allocation not successful.

The API throws the following exception error if the requested resource pool does not exist.

ResourceErrorException

The API throws the following exception error if the requested resource pool is exhausted.

AddressPoolException

The API throws the following exception error if the requested netmask is invalid.

InvalidNetmaskException

Verifying Responses for IP Allocations – Java APIs

Once the requesting service requests allocation through an API call, you can verify if the corresponding response is ready. The responses return the properties based on the request.

The following APIs help you to check if the response for the allocation request is ready.

Java API to check allocation request using cdb context

boolean com.tailf.pkg.ipaddressallocator.IPAddressAllocator. responseReady(NavuContext context, Cdb cdb, String poolName, String id)	API Parameters		
	Parameter	**Type**	**Description**
	Context	NavuContext	A NavuContext for the transaction
	Cdb	database	A database resource
	poolName	String	Name of the resource pool the request was created in
	id	String	Unique allocation ID for the allocation request

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

ready = IPAddressAllocator.responseReady(service.context(),cdb, poolName,
id);

returns True or False
```

Response

returns true if a response for the allocation is ready.

Java API to check allocation request without using cdb context

The following API is recommended to verify responses for IP allocations.

boolean com.tailf.pkg.ipaddressallocator.IPAddressAllocator. responseReady(NavuContext context, String poolName, String id)	API Parameters		
	Parameter	**Type**	**Description**
	Context	NavuContext	A NavuContext for the transaction
	poolName	String	Name of the resource pool the request was created in
	id	String	Unique allocation ID for the allocation request

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

ready = IPAddressAllocator.responseReady(service.context(), poolName,
id);
```

```
returns True or False
```

Response

returns true if a response for the allocation is ready.

Common exceptions raised by Java APIs for errors

- ResourceErrorException: if the allocation has failed, the request does not exist, or the pool does not exist
- ConfException: when there are format errors in the API request call
- IOException: when the I/O operations fail or are interrupted

Reading IP Allocation Responses for Java APIs

The following API reads the allocated IP subnet from the resource pool once the allocation request response is ready.

Subnet Read Java API to read allocation using cdb context

ConfIPPrefix com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRead(Cdb cdb, String poolName, String id)	API Parameters		
	Parameter	Type	Description
	cdb	Database	A database resource
	poolName	String	Name of the resource pool the request was created in
	id	String	Unique allocation ID for the allocation request

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

allocatedIP = IPAddressAllocator.subnetRead(cdb, poolName, id);


returns allocated IP subnet
```

Response

The API returns allocated subnet IP.

From Read Java API to read allocation using cdb context

ConfIPPrefix com.tailf.pkg.ipaddressallocator.IPAddressAllocator. fromRead(Cdb cdb, String poolName, String id)	API Parameters		
	Parameter	Type	Description
	cdb	Database	A database resource
	poolName	String	Name of the resource pool the request was created in
	id	String	Unique allocation ID for the allocation request

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

allocatedIP = IPAddressAllocator.fromRead(cdb, poolName, id);


returns allocated IP subnet
```

Response

Returns the subnet from which the IP allocation was made.

New Subnet Read Java API to read allocation

The following is the recommended API to read the allocated IP.

ConfIPPrefix com.tailf.pkg.ipaddressallocator.IPAddressAllocator. subnetRead(NavuContext context, String poolName, String id)	API Parameters		
	Parameter	Type	Description
	cdb	Database	A database resource
	poolName	String	Name of the resource pool the request was created in
	id	String	Unique allocation ID for the allocation request

Example

```
import com.tailf.pkg.ipaddressallocator.IPAddressAllocator;

allocatedIP = IPAddressAllocator.subnetRead(service.context(), poolName,
id);

returns allocated IP subnet
```

Response

Returns the allocated subnet for the IP.

Common exceptions raised by Java APIs for errors

- ResourceErrorException: if the allocation has failed, the request does not exist, or the pool does not exist
- ResourceWaitException: if the allocation is not ready

Using Python APIs for IP Allocations

Creating Python APIs for IP Allocations

The RM package exposes python APIs to manage allocation for IP subnet from resource pool. The below are the list of Python APIs exposed by RM package for

Default Python API for IP subnet allocation request
Description

The following API is used to create an allocation request for an IP address from a resource pool. Use the API definition **net_request** found in the module **resource_manager.ipaddress_allocator**.

The net_request function is designed to create an allocation request for a network. It takes several arguments, including the requesting service, username, pool name, allocation name, CIDR mask (size of the network), and optional parameters such as invert_cidr, redeploy_type, sync, and root. After calling this function, you need to call net_read to read the allocated IP from subnet

API def net_request (service, svc_xpath, username, pool_name, allocation_name, cidrmask, invert_cidr=False, redeploy_type="default", sync=False, root=None)	**API Parameters**		
	Parameter	**Type**	**Description**
	service		The requesting service node
	svc_xpath	String	xpath to the requesting service
	username	String	name of the user to use when redeploying the requesting service
	pool_name	Int	Name of the resource pool to make the allocation request from
	allocation_name	String	Unique allocation name
	cidrmask		Size of the network

	invert_cidr	Boolean	
	Redeploy_type		Service redeploy action. Available options: Default Touch Re-deploy Reactive-re-deploy
	Sync	Boolean	Allocation type, whether synchronous or asynchronous. By default, it is asynchronous.
	Root		Root node. If sync is set to true, you must provide a root node.

Example

```
import resource_manager.ipaddress_allocator as ip_allocator
    pool_name = "The Pool"
    allocation_name = "Unique allocation name"
    sync_alloc_name = "Unique synchronous allocation name"
    # This will try to asynchronously allocate the network of size 24 from the
pool named 'The Pool'
    # using allocation name: 'Unique allocation name'
    ip_allocator.net_request(service,
                             "/services/vl:loop-python[name='%s']" %
(service.name),
                             tctx.username,
                             pool_name,
                             allocation_name,
                             24)

    # This will try to synchronously allocate the network of size 24 from the
pool named 'The Pool'
    # using allocation name: 'Unique synchronous allocation name'
    ip_allocator.net_request(service,
                             "/services/vl:loop-python[name='%s']" %
(service.name),
                             tctx.username,
                             pool_name,
                             sync_alloc_name,
                             24,
                             sync=True,
                             root=root)
```

Python API for IP subnet allocation request with start IP address
Description

The following API is used to create a static allocation request for an IP address from a resource pool. Use the API definition **net_request_static** found in the module **resource_manager.ipaddress_allocator**.

The net_request_static function extends the functionality of net_request to allow for static allocation of network resources, specifically addressing individual IP addresses within a subnet. In addition to the parameters used in net_request, it also accepts subnet_start_ip, which specifies the starting IP address of the requested subnet. This function provides a way to allocate specific IP addresses within a network pool, useful for scenarios where certain IP addresses need to be reserved or managed independently. The function maintains similar error handling and package requirements as net_request, ensuring consistency in network resource management.

API def net_request_static(service, svc_xpath, username, pool_name, allocation_name, subnet_start_ip, cidrmask, invert_cidr=False, redeploy_type="default", sync=False, root=None)	**API Parameters**		
	Parameter	**Type**	**Description**
	service		The requesting service node
	svc_xpath	String	xpath to the requesting service
	username	String	name of the user to use when redeploying the requesting service
	pool_name	Int	Name of the resource pool to make the allocation request from
	allocation_name	String	Unique allocation name
	cidrmask		Size of the network
	invert_cidr	Boolean	
	Redeploy_type		Service redeploy action. Available options: Default Touch Re-deploy Reactive-re-deploy
	Sync	Boolean	Allocation type, whether synchronous or asynchronous. By default, it is asynchronous.
	Root		Root node. If sync is set to true, you must provide a root node.

Example

```
import resource_manager.ipaddress_allocator as ip_allocator
    pool_name = "The Pool"
    allocation_name = "Unique allocation name"
    sync_alloc_name = "Unique synchronous allocation name"
```

```
    # This will try to asynchronously allocate the address 10.0.0.8 from the
pool named 'The Pool'
    # using allocation name: 'Unique allocation name'
    ip_allocator.net_request_static(service,
                                    "/services/vl:loop-python[name='%s']" %
(service.name),
                                    tctx.username,
                                    pool_name,
                                    allocation_name,
                                    "10.0.0.8",
                                    32)

    # This will try to synchronously allocate the address 10.0.0.9 from the
pool named 'The Pool'
    # using allocation name: 'Unique synchronous allocation name'
    ip_allocator.net_request_static(service,
                                    "/services/vl:loop-python[name='%s']" %
(service.name),
                                    tctx.username,
                                    pool_name,
                                    sync_alloc_name,
                                    "10.0.0.9",
                                    32,
                                    sync=True,
                                    root=root)
```

Reading the allocated IP subnet once the allocation is ready.
Description

Use the API definition **net_read** found in the module **resource_manager.ipaddress_allocator** to read the allocated subnet IP.

The net_read function retrieves the allocated network from the specified pool and allocation name. It takes the username, root node for the current transaction, pool name, and allocation name as parameters. The function interacts with the rm_alloc module to read the allocated network, returning it if available or None if not ready. It's important to note that the function should be used to ensure that the response subnet is received in the current transaction, avoiding aborts or failures during commit.

API	API Parameters		
	Parameter	**Type**	**Description**
def net_read(username, root, pool_name, allocation_name)	username	String	name of the user to use when redeploying the requesting service
	Root		A maagic root for the current transaction
	pool_name	String	Name of the resource pool to make the allocation request from
	allocation_name	String	Unique allocation name

Example

```
# After requesting allocation, we check if ip is allocated.
    net = ip_allocator.net_read(tctx.username,
                                root,
                                pool_name,
                                allocation_name)
    if not net:
        self.log.info("Alloc not ready")
        return
    print ("net = %s" % (net))
```

ID Allocations

RM package exposes APIs to manage ID allocation from ID resource pool. The APIs are available to request ID, check if the allocation is ready and also to read the allocation once ready.

Using JAVA APIs for ID Allocations – Asynchronous Old APIs

The following are the asynchronous old Java APIs for ID allocation from RM resource pool.

Default Java API for ID allocation request
Description

The following API is used to create or update an ID allocation request with service redeploy type as default.

API			API Parameters
idRequest(NavuNode service, String poolName, String username, String id, boolean sync, long requestedId)	**Parameter**	**Type**	**Description**
	Service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the allocation ID from
	Username	String	name of the user to use when redeploying the requesting service
	ID	String	Unique allocation ID
	Sync	Boolean	sync allocations with the ID value across pools
	Requested ID	Int	Request the specific ID to be allocated

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

IdAllocator.idRequest(service, poolName, userName, id,
test_with_sync.booleanValue(), requestId);
```

Java API for ID allocation request with redeploy type
Description

The following API is used to create or update an IP allocation request with requesting service redeploy type as redeployType.

API			API Parameters
idRequest(NavuNode service, RedeployType redeployType, String poolName, String username, String id, boolean sync, long requestedId)	**Parameter**	**Type**	**Description**
	Service	NavuNode	NavuNode referencing the requesting service node
	redeployType		The available options are: Default Redeploytype Touch Reactive-re-deploy
	poolName	String	Name of the resource pool to request the allocation ID from
	Username	String	name of the user to use when redeploying the requesting service
	ID	String	Unique allocation ID
	Sync	Boolean	sync allocations with the ID value across pools
	Requested ID	Int	Request the specific ID to be allocated

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

IdAllocator.idRequest(service, redeployType, poolName, userName, id,
test_with_sync.booleanValue(), requestId);
```

Java API for ID allocation request with service context
Description

The following API is used to create or update an ID allocation request with requesting service redeploy type as default.

API			API Parameters
idRequest(ServiceContext context, NavuNode service, String poolName, String username, String id, boolean sync, long requestedId)	**Parameter**	**Type**	**Description**
	context	ServiceContext	Context referencing the requesting context that the service was invoked in
	service	NavuNode	NavuNode referencing the requesting service node
	poolName	String	Name of the resource pool to request the allocation ID from
	Username	String	name of the user to use when redeploying the requesting service
	ID	String	Unique allocation ID
	Sync	Boolean	sync allocations with the ID value across pools
	Requested ID	Int	Request the specific ID to be allocated

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

IdAllocator.idRequest(context, service, poolName, userName, id,
test_with_sync.booleanValue(), requestId);
```

Java API for ID allocation request with service context and redeploy type Description

Use the following API to create or update an ID allocation request with requesting service redeploy type as **redeployType**.

API	API Parameters		
idRequest(ServiceContext context, NavuNode service, RedeployType redeployType, String poolName, String username, String id, boolean sync, long requestedId)	**Parameter**	**Type**	**Description**
	context	ServiceContext	Context referencing the requesting context that the service was invoked in
	Service	NavuNode	NavuNode referencing the requesting service node
	RedeployType		Service redploy action. The available options are: Default Touch Re-deploy Reactive-re-deploy

	poolName	String	Name of the resource pool to request the allocation ID from
	Username	String	name of the user to use when redeploying the requesting service
	ID	String	Unique allocation ID
	Sync	Boolean	sync allocations with the ID value across pools
	Requested ID	Int	Request the specific ID to be allocated

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

IdAllocator.idRequest(context, service, redeployType, poolName, userName,
id, test_with_sync.booleanValue(), requestId);
```

Common exceptions raised by Java APIs for errors

The API may throw the below exception if no pool resource exists for the requested allocation.

ResourceErrorException

The API may throw the below exception if id request conflict with other allocation or does not match previous allocation in case of multiple owner request.

AllocationException

Verifying Responses for ID Allocations – Java APIs

RM package exposes responseReady Java API to verify if the ID allocation request is ready or not.

The following APIs are used to verify if the response is ready for ID allocation request.

Java API to check ID allocation ready using cdb context

API	API Parameters		
	Parameter	**Type**	**Description**
boolean responseReady(NavuContext context, Cdb cdb, String poolName, String id)	context	NavuContext	A NavuContext for the current transition
	poolName	Str	Name of the resource pool to request the allocation ID from
	cdb	database	The resource database
	id	String	Unique allocation ID

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

ready = IdAllocator.responseReady(service.context(), cdb, poolName, id);

returns True or False
```

Java API to check ID allocation ready without using cdb context

API	API Parameters		
boolean responseReady(NavuContext context, String poolName, String id)	**Parameter**	**Type**	**Description**
	NavuContext		A NavuContext for the current transition
	poolName	Str	Name of the resource pool to request the allocation ID from
	ID		Unique allocation ID

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

ready = IdAllocator.responseReady(service.context(), poolName, id);

returns True or False
```

Response

The API returns a true value if a response for the allocation is ready.

Common exceptions raised by Java APIs for errors

The API may throw the below exception if no pool resource exists for the requested allocation.

ResourceException

The API may throw the below exception when there are format errors in the API request call

ConfException

The API may throw the below exception when the I/O operations fail or are interrupted

IOException

Reading ID Allocation Responses for Java APIs

The following API reads information about specific allocation requests made by the API call. The response returns the allocated ID from the ID pool.

Java API to read ID allocation once ready using cdb context

Description

The following API is used to verify the response for an asynchronous ID allocation request.

API	API Parameters		
ConfUInt32 idRead(Cdb cdb, String poolName, String id)	**Parameter**	**Type**	**Description**
	cdb	Cdb	A database resource
	poolName	Str	Name of the resource pool to request the allocation ID from
	ID	String	Unique allocation ID

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

allocatedID = IdAllocator.idRead(cdb, poolName, id);

returns allocated ID
```

Java API to read ID allocation once ready without using cdb context

Description

The following API is used to verify the response for a synchronous ID allocation request.

API	API Parameters		
	Parameter	**Type**	**Description**
	context	NavuContext	A Navu context for the current transaction

ConfUInt32 idRead(NavuContext context, String poolName, String id)	poolName	String	Name of the resource pool to request the allocation ID from
	ID	String	Unique allocation ID

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

allocatedID = IdAllocator.idRead(service.context(), poolName, id);

returns allocated ID
```

Using JAVA APIs for ID Allocations – Synchronous/Asynchronous New APIs

The following are the synchronous/asynchronous new Java APIs exposed by RM package for ID allocation from resource pool.

Java API for ID allocation request using service context

Description

The following API is used to verify the response for a synchronous or an asynchronous ID allocation request.

API	API Parameters		
idRequest(ServiceContext context, NavuNode service, RedeployType redeployType, String poolName, String username, String id, boolean sync, long requestedId,boolean sync_alloc)	**Parameter**	**Type**	**Description**
	Context	ServiceContext	A context referencing the requesting context the service was invoked in
	service	NavuNode	Navu node referencing the requesting service node
	redeployType		Service redeploy action
	poolName	String	Name of the resource pool to request the allocation ID from
	Username	String	
	ID	String	Unique allocation ID
	Sync	Boolean	sync allocations with the ID value across pools
	requestedID	Int	A specific ID to be requested

	Sync_alloc	Boolean	If the boolean value is true, the allocation is synchronous

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

IdAllocator.idRequest(context, service, redeployType, poolName, userName,
id, test_with_sync.booleanValue(), requestId, syncAlloc.booleanValue());
```

Default Java API for ID allocation request

Description

The following API is used to verify the response for a synchronous or an asynchronous ID allocation request.

API	API Parameters		
	Parameter	**Type**	**Description**
idRequest(NavuNode service, RedeployType redeployType, String poolName, String username, String id, boolean sync, long requestedId,boolean sync_alloc)	service	NavuNode	Navu node referencing the requesting service node
	redeployType		Service redeploy action. The available options are: Default Touch Re-deploy Reactive-re-deploy
	poolName	String	Name of the resource pool to request the allocation ID from
	Username	String	
	ID	String	Unique allocation ID
	Sync	Boolean	sync allocations with the ID value across pools
	requestedID	Int	A specific ID to be requested
	Sync_alloc	Boolean	Synchronous allocation

Example

```
import com.tailf.pkg.idallocator.IdAllocator;

IdAllocator.idRequest(service, redeployType, poolName, userName, id,
test_with_sync.booleanValue(), requestId, syncAlloc.booleanValue());
```

Common exceptions raised by Java APIs for errors

The API may throw the below exception if no pool resource exists for the requested allocation.

ResourceErrorException

The API may throw the below exception if id request conflict with other allocation or does not match previous allocation in case of multiple owner request.

AllocationException

Using Python APIs for ID Allocations

RM package also exposed Python APIs to request ID allocation from a resource pool. The below APIs are Python APIs exposed by RM for ID allocation

Python API for default ID allocation request
Description

Use the module **resource_manager.id_allocator**.

The id_request function is used to create an allocation request for an ID. It takes several arguments including the service, service xpath, username, pool name, allocation name, sync flag, requested ID (optional), redeploy type (optional), alloc sync flag (optional), and root (optional).

API	API Parameters		
	Parameter	Type	Description
id_request(service, svc_xpath, username, pool_name, allocation_name, sync,requested_id=-1 ,redeploy_type="default", alloc_sync=False, root=None):	service		The requesting service node
	svc_xpath	Str	xpath to the requesting service
	username	Str	name of the user to use when redeploying the requesting service
	pool_name	Str	Name of the resource pool to make the allocation request from
	allocation_name	Str	Unique allocation name

	Sync	Boolean	Sync allocations with this name across the pool
	Requested Id	Int	A specific ID to be requested
	Redeploy_type		Service redeploy action. Available options: Default Touch Re-deploy Reactive-re-deploy
	alloc_sync	Boolean	Allocation type, whether synchronous or asynchronous. By default, it is asynchronous.
	root		root node. If sync is set to true, you must provide a root node.

Example

```
import resource_manager.id_allocator as id_allocator
    pool_name = "The Pool"
    allocation_name = "Unique allocation name"

    # This will try to allocate the value 20 from the pool named 'The Pool'
    # using allocation name: 'Unique allocation name'
    # It will allocate the id asynchronously from the pool ‘The Pool’
    id_allocator.id_request(service,
                            "/services/vl:loop-python[name='%s']" %
(service.name),
                            tctx.username,
                            pool_name,
                            allocation_name,
                            False,
                            "firstfree",
                            20)
    #The below will allocate the id synchronously from the pool ‘The Pool’
    id_allocator.id_request(service,
                            "/services/vl:loop-python[name='%s']" %
(service.name),
                            tctx.username,
                            pool_name,
                            allocation_name,
                            True,
                            "firstfree",
                            20)



     vlan_id = id_allocator.id_read(tctx.username, root, 'vlan-pool', service.name)

     if vlan_id is None:
          self.log.info(f"Allocation not ready...")
          return propList
```

```
    self.log.info(f"Allocation is ready: {vlan_id}")

    service.vlan_id = vlan_id
```

Python API to read allocated ID once the allocation is ready

Description

Use the API definition id_read found in the module **resource_manager.id_allocator** to read the allocated ID.

The `id_read` function is designed to return the allocated ID or None if the ID is not yet available. It first tries to look up the ID in the current transaction using the provided `root` and `pool_name` and `allocation_name`. If the ID is available in the current transaction, it returns the ID. If there is an error, it raises a `LookupError`. If the ID is not available in the current transaction, it calls `id_read_async` to asynchronously retrieve the ID.

API	API Parameters		
	Parameter	**Type**	**Description**
id_read(username, root, pool_name, allocation_name)	username	Str	name of the user to use when redeploying the requesting service
	Root		A maagic root for the current transaction
	pool_name	Str	Name of the resource pool to make the allocation request from
	allocation_name	Str	Unique allocation name

Example

```
# After requesting allocation, we check if the allocated ID is available

id = id_allocator.id_read(tctx.username, root,
                          pool_name, allocation_name)
    if not id:
        self.log.info("Alloc not ready")
        return
     print ("id = %d" % (id))
```