



**Implementation Agreement for  
Network Resource Control Protocol (NRCP)**

**MSF-IA-NRCP.001-FINAL**

# Multiservice Switching Forum Implementation Agreement

**Contribution Number:** msf2004.118.02

**Document Filename:** MSF-IA-NRCP.001-FINAL

**Working Group:** Protocol and control

**Title:** Implementation agreement for Network Resource Control Protocol (NRCP)

**Editors:**

|                         |                            |                              |
|-------------------------|----------------------------|------------------------------|
| Olov Schelén            | Anders Torger              | Joachim Johansson            |
| Olov.Schelen@operax.com | Anders.Torger@operax.com   | Joachim.Johansson@operax.com |
| Ulf Bodin               | Håkan Lennestål            |                              |
| Ulf.Bodin@operax.com    | Hakan.Lennestal@operax.com |                              |

**Working Group Chairperson:** Chris Gallon

**Date:** 2004-09-29

**Abstract:** The Network Resource Control Protocol is designed for communication with Bandwidth Managers. It is a general-purpose object-based protocol for requesting network resources (bandwidth) for various application data streams and traffic aggregates. The objects can be carried with a standard container protocol (e.g., XML, COPS, SNMP, CORBA). Client side implementations of the protocol can provide a standard program interface (e.g., Java, C++) to allow simple software integration. Clients include application frameworks/servers, application signalling proxies, other bandwidth managers, etc.

This document describes the information model for interaction with bandwidth managers and instantiates the model by defining a Java-based client-side interface.

**Keywords:** Bandwidth Manager, Bandwidth Broker, QoS control

The goal of the MSF is to promote multi-vendor interoperability as part of a drive to accelerate the deployment of next generation networks. To this end the MSF looks to adopt pragmatic solutions in order to maximize the chances for early deployment in real world networks.

To date the MSF has defined a number of detailed Implementation Agreements and detailed Test Plans for the signaling protocols between network components and is developing additional Implementation Agreements and Test Plans addressing some of the other technical issues such as QoS and Security to assist vendors and operators in deploying interoperable solutions.

In 2002, the MSF held a "Global MSF Interoperability 2002" (GMI 2002) event that tested interoperability between next generation network elements situated in Asia, Europe and North America. GMI 2002 validated the MSF release 1 architectural framework and Implementation Agreements by subjecting them to interoperability testing based on realistic network scenarios.

Following the success of GMI 2002 the MSF work program continues to address the key technical barriers to next generation network deployments. Global MSF Interoperability 2004 (GMI 2004) will demonstrate a deployable and operationally ready IP telephony network with Network Management, enhanced Quality-of-Service (QoS) and security features. GMI2004 will also demonstrate a service layer with application server, media server, and service broker functionality. This will enable the MSF to demonstrate a full end-to-end customer ready deployable network.

It is envisaged that GMI2004 will provide an industry showcase that will:

- Assist carriers achieve their goal: to deploy flexible, best of breed products.
- Assist vendors achieve their goal: to market products more cost effectively.
- Display the global interoperability of the MSF architecture as referenced in the Release 2 architecture document.
- Demonstrate a network scenario that can be managed to specific quality standards.

The MSF welcomes feedback and comment and would encourage interested parties to get involved in this work program. Information about the MSF and membership options can be found on the MSF website <http://www.msforum.org/>

## DISCLAIMER

The information in this publication is believed to be accurate as of its publication date. Such information is subject to change without notice and the Multiservice Switching Forum is not responsible for any errors or omissions. The Multiservice Switching Forum does not assume any responsibility to update or correct any information in this publication. Notwithstanding anything to the contrary, neither the Multiservice Switching Forum nor the publisher make any representation or warranty, expressed or implied, concerning the completeness, accuracy, or applicability of any information contained in this publication. No liability of any kind whether based on theories of tort, contract, strict liability or otherwise, shall be assumed or incurred by the Multiservice Switching Forum, its member companies, or the publisher as a result of reliance or use by any party upon any information contained in this publication. All liability for any implied or express warranty of merchantability or fitness for a particular purpose is hereby disclaimed.

The receipt or any use of this document or its contents does not in any way create by implication or otherwise:

Any express or implied license or right to or under any Multiservice Switching Forum member company's patent, copyright, trademark or trade secret rights which are or may be associated with the ideas, techniques, concepts or expressions contained herein; nor

Any warranty or representation that any Multiservice Switching Forum member companies will announce any product(s) and/or service(s) related thereto, or if such announcements are made, that such announced product(s) and/or service(s) embody any or all of the ideas, technologies, or concepts contained herein; nor

Any commitment by a Multiservice Switching Forum company to purchase or otherwise procure any product(s) and/or service(s) that embody any or all of the ideas, technologies, or concepts contained herein; nor

Any form of relationship between any Multiservice Switching Forum member companies and the recipient or user of this document.

Implementation or use of specific Multiservice Switching Forum Implementation Agreements, Architectural Frameworks or recommendations and Multiservice Switching Forum specifications will be voluntary, and no company shall agree or be obliged to implement them by virtue of participation in the Multiservice Switching Forum.

For addition information contact:

Multiservice Switching Forum  
39355 California Street, Suite 307, Fremont, CA 94538  
(510) 608-5922  
(510) 608-5917 (fax)  
[info@msforum.org](mailto:info@msforum.org)  
<http://www.msforum.org>

## Table of Contents

|       |   |    |
|-------|---|----|
| 1.    | <a href="#">Scope of this Document</a>  | 5  |
| 2.    | <a href="#">Definitions</a>   | 5  |
| 3.    | <a href="#">The Bandwidth Manager</a>   | 5  |
| 3.1   | <a href="#">Admission control metrics: bit-rate and network service class</a> | 6  |
| 3.2   | <a href="#">Resource partitions and call priority</a>                         | 7  |
| 3.3   | <a href="#">Peering and inter-domain call routing</a>                         | 7  |
| 3.4   | <a href="#">Multiple BMs in a network</a>                                     | 7  |
| 3.5   | <a href="#">Inter-domain BM peering</a>                                       | 7  |
| 4.    | <a href="#">NRCP Protocol Description</a>                                     | 8  |
| 4.1   | <a href="#">Protocol characteristics</a>                                      | 8  |
| 4.2   | <a href="#">Requirements for the Bearer Protocol</a>                          | 8  |
| 4.3   | <a href="#">Two-phase and single phase resource allocation</a>                | 9  |
| 4.4   | <a href="#">Header Information</a>  | 9  |
| 4.5   | <a href="#">Two-phase signalling diagram example</a>                          | 10 |
| 4.6   | <a href="#">Single-phase signalling diagram example</a>                       | 11 |
| 4.7   | <a href="#">Common parameter sets</a>   | 11 |
| 4.7.1 | <a href="#">Resource specification</a>  | 11 |
| 4.7.2 | <a href="#">Filter specification</a>  | 13 |
| 4.7.3 | <a href="#">NAT specification and NAT traversal</a>                           | 14 |
| 4.8   | <a href="#">Error message</a>   | 15 |
| 4.9   | <a href="#">Reserve Bandwidth Request (RESV_REQ)</a>                          | 15 |
| 4.10  | <a href="#">Reserve Bandwidth Reply (RESV_REPLY)</a>                          | 16 |
| 4.11  | <a href="#">Commit Reservation Request (COMMIT_REQ)</a>                       | 17 |
| 4.12  | <a href="#">Commit Reservation Reply (COMMIT_REPLY)</a>                       | 17 |
| 4.13  | <a href="#">Reserve-and-Commit Request (RESVCM_REQ)</a>                       | 18 |
| 4.14  | <a href="#">Reserve-and-Commit Reply (RESVCM_REPLY)</a>                       | 19 |
| 4.15  | <a href="#">Modify Reservation Request (MODIFY_REQ)</a>                       | 19 |
| 4.16  | <a href="#">Modify Reservation Reply (MODIFY_REPLY)</a>                       | 20 |
| 4.17  | <a href="#">Release Reservation Request (RELEASE_REQ)</a>                     | 20 |
| 4.18  | <a href="#">Release Reservation Reply (RELEASE_REPLY)</a>                     | 20 |
| 4.19  | <a href="#">Reservation Termination Callback (TERM_CB)</a>                    | 20 |
| 4.20  | <a href="#">Reservation Get Request (GET_REQ)</a>                             | 21 |
| 4.21  | <a href="#">Reservation Get Reply (GET_REPLY)</a>                             | 21 |
| 5.    | <a href="#">Appendix A – NRCP Java API</a>                                    | 23 |

## 1. Scope of this Document

This document describes the information model for the IF-2 and IF-5 interfaces (figure 1) to the bandwidth manager (BM). In addition, a Java-based application programming interface (API) is presented. A BM may provide this client-side API for integration (e.g., with call managers or other application frameworks).

The protocol between the client-side API and the BM is called NRCP, Network Resource Control Protocol. The carrier protocol for NRCP could for example be XML, COPS, SNMP, CORBA. Such protocol instantiations may be described in separate documents.

The IF-2 and IF-5 interfaces, used for call control functions and inter-BM communication respectively, are defined in msf2003.002.

Apart from the mandatory features necessary for basic operation, this document also describes optional features that may be used for extended services and to improve scalability in terms of aggregation.

## 2. Definitions

BM     Bandwidth Manager  
CA     Call Agent

## 3. The Bandwidth Manager

A Bandwidth Manager (BM), also known as a Network Resource Controller (NRC) or a QoS Controller, is a software component for dynamic QoS control in IP networks. In each operator domain there may be a single BM or many distributed BMs constituting a network resource control plane.

The network is typically configured for differentiated forwarding (using Diffserv, MPLS, specific layer two mechanisms, etc). The BM provides admission control for various forwarding classes/resources and may gate-keep resources by configuring traffic conditioners (filters, token bucket shapers and markers).

Since differentiated forwarding can be implemented with several network mechanisms, the notion of a *network service class* is introduced to characterise resources and resource requests. Access to a network service class is controlled by the BM. Depending on the availability of resources a BM will either admit or deny bandwidth requests. Thus, by controlling the access to certain forwarding resources a BM can dynamically provide differentiated IP services with predictable network transport quality.

The scope of this document is to define the protocol for bandwidth request with a BM (IF-2 and IF-5). Note that the same protocol is used independently of whether the request is for a specific application data stream (often occurring at IF-2) or for a bandwidth aggregate (often occurring at IF-5).

As seen in the market, bandwidth manager products vary greatly in what features they support. While providing the necessary functionality, IF-2 and IF-5 should be simple enough to be compatible with BMs with limited feature sets. In order not to limit more advanced BMs, extended features are supported by adding optional parameter sets to the protocol.



### **3.2 Resource partitions and call priority**

Resources within a Network Service Class can be partitioned for various purposes (e.g., normal calls, emergency calls, gold calls etc) or for various retail players in a network wholesale scenario. The partitions are used for admission control only and are typically not mapped into partitions in the forwarding plane (although such mapping may exist in some cases). Partitions can be chained by internal policies in various ways so that resources for specific purposes can be exclusive, limited, overlapping etc. Partitions are managed in the BM by the operator. Through NRCP, application servers can indicate from which partition resources for individual request should be allocated.

There is also a notion of call priority. This can be used to give relative priority to some calls before others within partitions. Top priority calls may pre-empt other calls. Priority can be used to ensure that calls are served.

As an example, there can be a partition where resources are ensured (and/or limited) for emergency calls so that they can normally be served without pre-emption. In addition, call priority can be used to indicate that the emergency call may pre-empt other calls (if needed).

### **3.3 Peering and inter-domain call routing**

In the MSF architecture, inter-domain peering is provided at the application signalling (SIP) level (IF-6). In this model, call agents are responsible for call routing between domains. For this, call agents rely on Session Border Control (SBC) functionality at the network boundaries to steer traffic to selected egress points. In addition, the call agent will ensure that the SIP signalling crosses the same network boundary points as the payload on a per-call basis. Inside each domain a per-call request for bandwidth is performed with the local BM (using IF-2).

### **3.4 Multiple BMs in a network**

In order to scale bandwidth management to handling individual calls in the backbone it is possible to deploy multiple BMs in a single network. E.g., to manage separate resource objects of the network with different BMs and to handle high numbers of busy hour call attempts (BHCA) with multiple BMs.

IF-5 is used for communication between multiple BMs in a network. This interface can be used to deploy multiple top-tier BMs that serve per-call admission requests. Top-tier BMs can allocate aggregate resources from bottom-tier BMs that manage the network resource objects.

### **3.5 Inter-domain BM peering**

IF-5 could also be used to inter-link BMs over multiple domains. This model (which is not the currently proposed MSF model) is referred to as BM *peering* and enables multi-domain resources. IF-5 is then used for communication between BMs located in different domains.

Inter-domain signalling can be minimised by pre-allocating aggregate inter-domain resources to frequently used destinations. Thus if a BM entity processing a resource request finds out that the source address is local and the destination lies in another domain, that entity can do one of two things:

- Generate a request to the BM peer in charge of the particular destination and wait for a response.
- Use previously negotiated resources to that destination and respond to the client without involving the BM peer.

The BM peering model means that core BMs handle aggregate reservations instead of per-call reservations.

## 4. NRCP Protocol Description

In the following, the NRCP protocol is described in detail concerning what type of information that needs to be transported. In appendix A, a Java-based client-side interface according to this information model is defined. An exact specification of message layout and binding to a bearer protocol for transport between the client-side interface and a BM is not provided in this document.

NRCP is always client/server oriented. When used in IF-2, the BM acts as the server, and the other end (for example a SIP proxy) is the client. When used in IF-5 for multi BM deployments in a single domain, upper level BMs act as clients communicating with lower level BMs. For full peering over IF-5, there are two protocol connections, one in each direction, where each BM acts server on one connection and client on the other.

The protocol is split in one mandatory and one optional part. The mandatory part must be implemented by all entities supporting the protocol. If a server receives a request including unsupported parameters it must drop the message and reply with an error message. Clients on the other hand, should handle as many of the message parameters as possible and simply ignore any unrecognised parts of the message.

How the presence of any optional parameters in a message is indicated is part of the specification of the specific bearer protocol or API. Optional parameters not included may be indicated with zero or null values as in the Java-based API presented in appendix A. Other examples of indicating the presence of optional parameters includes existence of specific TLV:s or adding specific flags for each optional parameter.

### 4.1 Protocol characteristics

The NRCP protocol has the following notable characteristics:

- Message-based client/server where a BM is the server.
- Client requests are answered with server replies.
- Unsolicited messages may be sent from the server to the client.
- Each message is handled atomically.
- Address parameters are IP addresses. It is mandatory to support IPv4 addresses, and optional to support IPv6 addresses.

### 4.2 Requirements for the Bearer Protocol

The bearer protocol on which NRCP is implemented must support the following specific features:

- Unsolicited messages from server to client

The following is not strictly necessary, although strongly recommended:

- Authentication and Authorization.
- Secure communication

### 4.3 Two-phase and single phase resource allocation

NRCP supports two-phase and single-phase resource allocation. In the two-phase case, resources are first reserved in the BM, which only ensures that resources are available. When the reservation later is committed with a separate message, the resource can be implemented in the network if needed (typically setting up traffic conditioners in edge routers/switches).

Two-phase resource allocation is typically used for voice/video calls, where resources are reserved before ringing in the remote end, and committed when the callee goes off-hook. In this case, full filter information needed for cutting-through the call may not be known until the callee answers. The exact amount of resources may also be unknown at the reserve phase, since the calling parts may not have agreed on codec yet. In such cases, the amount required of a default codec is typically reserved, and the reservation bandwidth is then optionally modified in the commit phase. The purpose of the initial reserve phase in this example is to provide a guarantee that there are resources available for the call using a default codec before ringing starts.

Another use of two-phase resource allocation is for making reservations in advance. This can for example be used to reserve resources for a scheduled video conference, where one wants to make sure in advance that resources will be available, but filters are not known until the video clients join in at the start.

Single-phase resource allocation is used whenever all parameters are known when the resource request is to be made. This minimises the setup time.

### 4.4 Header Information

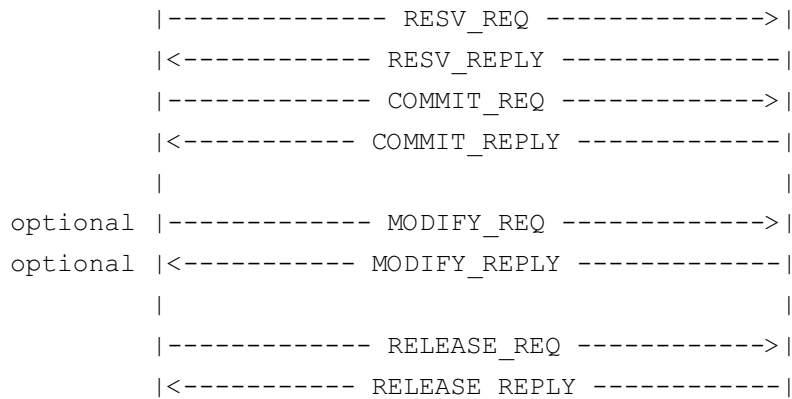
| Name         | Description   | Mandatory/<br>Optional |
|--------------|---|------------------------|
| Version      | Protocol version  | Mandatory              |
| Message id   | Client-specified message identifier, which is kept in the reply. Makes it possible to pipeline several requests while being able to multiplex the replies (which may be out of order).<br><br>The message id must be at least 16 bits wide to provide a proper pipeline space.  | Mandatory              |
| Message type | The message type indicates the overall purpose of the message.<br><br><ol style="list-style-type: none"> <li>1. Reserve bandwidth request (RESV_REQ)</li> <li>2. Reserve bandwidth reply (RESV_REPLY)</li> <li>3. Commit reservation request (COMMIT_REQ)</li> <li>4. Commit reservation reply (COMMIT_REPLY)</li> <li>5. Reserve-and-commit request (RESVCM_REQ)</li> <li>6. Reserve-and-commit reply (RESVCM_REPLY)</li> <li>7. Modify reservation request (MODIFY_REQ)</li> <li>8. Modify reservation reply (MODIFY_REPLY)</li> <li>9. Release reservation request (RELEASE_REQ)</li> <li>10. Release reservation reply (RELEASE_REPLY)</li> <li>11. Reservation termination callback (TERM_CB)</li> <li>12. Reservation get request (GET_REQ)</li> <li>13. Reservation get reply (GET_REPLY)</li> </ol> | Mandatory              |

|                     |   |           |
|---------------------|---|-----------|
|                     | The message type field size should allow for at least 255 message types.  |           |
| Options             | 32 bit general purpose field which supplements the message type. Currently there are no specified uses for this field.  | Mandatory |
| Payload type        | Type of payload carried by the message.<br>1. Reservation parameters.<br>2. Error message.<br><br>The payload type field size should allow for at least 255 types.  | Mandatory |
| Parameter set count | Number of parameter sets in the message. 0 – 65535 sets.<br><br>Some commands can carry more than one instance of its associated parameter set, for example a RESV_REQ can contain a request for more than one bandwidth reservation. | Mandatory |
| Session id          | Identifier to identify a single session carried in the protocol. Allows multiple authorized sessions on a single connection.  | Optional  |
| Timestamp           | Time when the message was sent. Can be used for diagnostic and/or security mechanisms.  | Optional  |

#### 4.5 Two-phase signalling diagram example

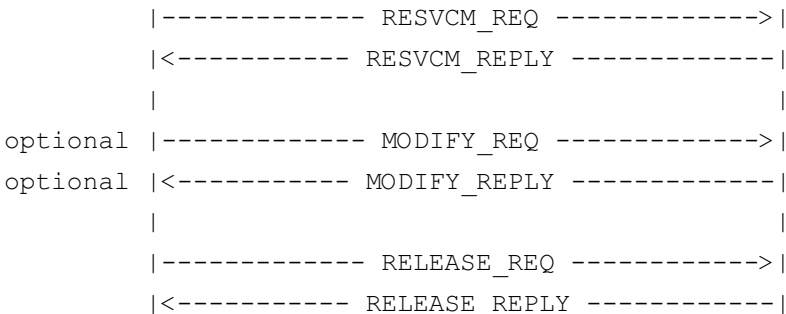
The signalling diagram below shows the normal course of events for a two-phase reservation, with an optional modify request. The client is to the left; the server (BM) is to the right.

Should the BM reply with an error message in any stage, the reservation will have its previous state. Example: if the COMMIT\_REQ fails, the state will remain the same as before (i.e., as reported from the previous RESV\_REPLY). If a client wants to revert further than that, it must still release the reservation.



#### 4.6 Single-phase signalling diagram example

The signalling diagram below shows the normal course of events for a single-phase reservation, with an optional modify request.



#### 4.7 Common parameter sets

The information in the different requests can be grouped into two groups. First the resource specification, which defines the amount of network resources that are to be reserved and then the filter specification, which adds information about the specific traffic flow that is to receive the network resource. These two specifications are typically used in the two-phase allocation model where the resource specification is first being used to specify the desired service and to reserve necessary resources through the network. The filter specification is used in the succeeding commit phase to specify the traffic that should obtain the service requested in the first phase. For modification and single-phase allocation both the resource and the filter specifications are given in the same message.

In addition to the resource and filter specifications an ID is used to identify a reservation. This ID is used in the commit phase, when modifying or releasing a reservation. The ID is returned from a RESV\_REQ or from the single-phase RESVCM\_REQ. It is assigned by the BM and is unique within a BM.

| Name | Description  | Mandatory/<br>Optional |
|------|--|------------------------|
| ID   | A 32 bit ID assigned by the BM to identify the reservation | Mandatory              |

##### 4.7.1 Resource specification

The resource specification contains information for the BM to perform admission control. The bandwidth parameter specifies the amount of network resources, the network service class specifies which kind of resources to reserve, and the addresses specifies from where to where. Priority is used for call pre-emption and emergency calls. There may also be a number of optional parameters (e.g., start and stop times) which specify the reservation further.

| Name                  | Description  | Mandatory/<br>Optional |
|-----------------------|--|------------------------|
| Bandwidth             | An unsigned 32 bit integer which contains the requested amount of bandwidth in kilobits per second.  | Mandatory              |
| Network service class | Type of (network transport) service. The classes are defined in separate specifications. A network service class could for example be conversational, streaming, | Mandatory              |

## Implementation agreement for Network Resource Control Protocol (NRCP)

|                              |   |           |
|------------------------------|---|-----------|
|                              | interactive, or data. The network service class indicates to the BM how to interpret the bandwidth parameter and how to set up traffic conditioner parameters.  |           |
| Priority                     | <p>An unsigned 8 bit integer defining the priority of the reservation, where a higher number represents a higher priority.</p> <p>The highest bit is the pre-emption bit, which is mandatory to support. When the pre-emption bit is set, the bandwidth request should pre-empt lower prioritised reservations if necessary.</p> <p>Fine-granular priority differentiation in the ranges 0 – 127 and 128 – 255 is optional.</p> | Mandatory |
| Source address               | Together with the source mask length, the source address specifies the source of the reservation.   | Mandatory |
| Source mask length           | <p>If not set, the full source address is used to specify the source. If set, the source address parameter is masked with the mask length to form a source prefix.</p> <p>The purpose of specifying a source prefix instead of a source address, is to reserve for all hosts within a subnet rather than a single host. May be used for aggregation.</p>  | Optional  |
| Destination address          | Together with the destination mask length, the destination address specifies the destination of the reservation.  | Mandatory |
| Destination mask length      | The mask length of the destination address. This can be used for aggregation.   | Optional  |
| VPN ID                       | If MPLS VPNs are used, this ID identifies in which VPN the traffic flow will reside.  | Optional  |
| Minimal acceptable bandwidth | Typically used to indicate what the least demanding CODEC requires. If the reservation is associated to a combined video and voice call, the bandwidth parameter is typically set to what is required by the combination, and the minimal acceptable bandwidth parameter is set to the voice-only requirement.  | Optional  |
| Partition                    | <p>If the resources are partitioned between users and/or applications within a network service class, this parameter will indicate which partition to deduct the bandwidth from.</p> <p>Partitions are typically used to dedicate resources to prioritized users.</p>   | Optional  |
| Ingress address              | <p>For inter-domain bandwidth requests, the client BM can specify where the traffic will enter the domain controlled by the server BM.</p> <p>It may also be used in cases when the client knows the ingress, while the BM cannot calculate it from the source</p>  | Optional  |

## Implementation agreement for Network Resource Control Protocol (NRCP)

|                                 |   |          |
|---------------------------------|---|----------|
|                                 | address.  |          |
| Egress address                  | May be used in cases when the client knows the egress, while the BM cannot calculate it from the destination address.   | Optional |
| Egress mode                     | The egress address may be interpreted either as a requirement or as a hint. In case of a requirement, the reservation will fail if the requirement cannot be met. <ol style="list-style-type: none"> <li>1. Hint</li> <li>2. Requirement</li> </ol>                         | Optional |
| Additional source prefixes      | For VPN use. To be specified.   | Optional |
| Additional destination prefixes | For VPN use. To be specified.   | Optional |
| Start time                      | Start time for an reservation in advance. UTC, smallest unit is seconds.  | Optional |
| Stop time                       | Stop time for a time-limited reservation. UTC, smallest unit is seconds.  | Optional |
| Time options                    | Specifies how to interpret start and stop times. <ol style="list-style-type: none"> <li>1. Open-ended reservation.</li> <li>2. Times are specified relative from now.</li> <li>3. Times are specified in absolute values.</li> <li>4. Periodicity, to be defined</li> </ol> | Optional |
| User                            | A string identifier to associate the request to a user found in an external system.   | Optional |

#### 4.7.2 Filter specification

When resources are reserved additional filter parameters are required to further specify which individual flow that is to receive the requested network service. The source and destination addresses are already included in the resource specification so the additional information needed to identify a flow is source and destination ports and protocol byte. A diffserv code point can also be used to identify a flow. Note that this information is used for traffic filtering/conditioning at ingress. The implementation of the service in the network in terms of diffserv, MPLS or other mechanisms is part of the operator internal policies for the requested network service class.

| Name                 | Description   | Mandatory/Optional |
|----------------------|---|--------------------|
| Source port          | The source port of the traffic flow associated to the reservation. Will typically be used when installing traffic conditioner parameters. | Mandatory          |
| Destination port     | The destination port of the traffic flow associated to the reservation.   | Mandatory          |
| Protocol             | The IP header protocol number (defined by IANA) of the traffic flow associated to the reservation.  | Mandatory          |
| Source port low      | Forms a port range together with the source port parameter.   | Optional           |
| Destination port low | Forms a port range together with the destination port parameter.  | Optional           |

|      |  |          |
|------|--|----------|
| DSCP | DiffServ Code Point of the traffic flow associated to the reservation. | Optional |
|------|--|----------|

#### 4.7.3 NAT specification and NAT traversal

In an MSF network, source and destination addresses in reservations (sect 6.7.1) refer to inside (private) addresses of ingress and egress points of the operator network. The ports given in filters (sect 6.7.2) refer to ports used inside the private address space. In an MSF network, addresses and ports are fully specified (i.e., no address masks and port ranges are used).

NRCP can provide BMs with information to set up NAT mappings at ingress and egress points of the operator network. In addition to the address and port information for the inside (as specified above) this information includes address and port information for incoming traffic at ingress and for outgoing traffic at egress. Consequently, through separate ingress and egress translations, three segments are supported: before arriving to ingress, inside the operator network, and after leaving the egress of operator network. In some cases addresses and ports for transit traffic leaving the egress are restored to the same values as when it arrived at ingress, but in the general case addresses/ports may be different.

| Name                 | Description   | Mandatory /Optional |
|----------------------|---|---------------------|
| NAT in from address  | Source address of traffic arriving to the operator ingress      | Optional            |
| NAT in from port     | Source port of traffic arriving to the operator ingress         | Optional            |
| NAT in to address    | Destination address of traffic arriving to the operator ingress | Optional            |
| NAT in to port       | Destination port of traffic arriving to the operator ingress    | Optional            |
| NAT out from address | Source address of traffic leaving the operator egress           | Optional            |
| NAT out from port    | Source port of traffic leaving the operator egress              | Optional            |
| NAT out to address   | Destination address of traffic leaving the operator egress      | Optional            |
| NAT out to port      | Destination port of traffic leaving the operator egress         | Optional            |

If the external addresses at either (or both) ingress and egress points are unknown to the client then NAT traversal functions are required. This is indicated by the NAT traversal parameter, which if present, informs the bandwidth manager that the ingress and/or egress node must be configured for NAT traversal. If NAT traversal is indicated at the ingress for a flow the in from address and port parameters are not significant (as they are unknown), the in to address and port parameters are significant. If NAT traversal is indicated at the egress for a flow then the out to address and port parameters are not significant (as they are unknown), the out from address and port parameters are significant.

Notice that NRCP supports bi-directional reservations by sending two linked uni-directional reservations within the same RESV\_REQ. Therefore if a given reservation has, for example, NAT traversal indicated at the ingress then the other reservation received within the RESV\_REQ must have NAT traversal indicated at the egress. This is required because NAT traversal will result in the edge node deriving the IP addresses and ports for the egress flow from the IP addresses and ports seen in the ingress flow.

| Name          | Description   | Mandatory/Optional |
|---------------|---|--------------------|
| NAT traversal | If specified NAT traversal is required but the NAT mappings are unknown. <ol style="list-style-type: none"> <li>1. NAT traversal at both ingress and egress</li> <li>2. NAT traversal at ingress only</li> <li>3. NAT traversal at egress only</li> </ol> | Optional           |

#### 4.8 Error message

Should a request not be completed due to some error, the server will reply with an error message. Note that the message type field in the header is the same as on a normal reply. The payload type indicates if the reply is an error or not.

| Name         | Description  | Mandatory/Optional |
|--------------|--|--------------------|
| Error Code   | An error code representing the given error. <ol style="list-style-type: none"> <li>1. Parameter set not supported. This error code is returned by the server if the given parameter set is unsupported.</li> <li>2. Reservation denied.</li> <li>3. BM unreachable</li> </ol> Further error codes to be defined. | Mandatory          |
| Error String | A textual string describing the error in human-readable form   | Mandatory          |

#### 4.9 Reserve Bandwidth Request (RESV\_REQ)

The reserve bandwidth request is sent from the client to the server when the client wants to reserve resources. The BM must be able to handle at least two parameter sets, that is at least two bandwidth reservations per request. The purpose of this is to allow for an atomic bi-directional bandwidth reservation request.

| Name                         | Mandatory/Optional |
|------------------------------|--------------------|
| Bandwidth                    | Mandatory          |
| Network service class        | Mandatory          |
| Priority                     | Mandatory          |
| Source address               | Mandatory          |
| Source mask length           | Mandatory          |
| Destination address          | Mandatory          |
| Destination mask length      | Mandatory          |
| VPN ID                       | Optional           |
| Minimal acceptable bandwidth | Optional           |
| Partition                    | Optional           |
| Ingress address              | Optional           |

|                                 |          |
|---------------------------------|----------|
| Egress address                  | Optional |
| Egress mode                     | Optional |
| Additional source prefixes      | Optional |
| Additional destination prefixes | Optional |
| Start time                      | Optional |
| Stop time                       | Optional |
| Time options                    | Optional |
| User                            | Optional |
| Source port                     | Optional |
| Destination port                | Optional |
| Protocol                        | Optional |
| Source port low                 | Optional |
| Destination port low            | Optional |
| DSCP                            | Optional |
| NAT in from address             | Optional |
| NAT in from port                | Optional |
| NAT in to address               | Optional |
| NAT in to port                  | Optional |
| NAT out from address            | Optional |
| NAT out from port               | Optional |
| NAT out to address              | Optional |
| NAT out to port                 | Optional |
| NAT traversal                   | Optional |

#### **4.10 Reserve Bandwidth Reply (RESV\_REPLY)**

The reply contains the same amount of parameter sets as the associated request (which is atomic). Only the input parameters that may have been changed by the BM are included in the reply, plus an ID which identifies the reservation for further access (commit/modify/release). One parameter set is returned for each parameter set in the request. If bandwidth-range is supported the admitted bandwidth is returned and if generalization of network prefixes are supported the generalized prefixes are returned. The parameter sets in the request are treated atomically, so if any of them cannot be admitted, an error message will be returned.

| <b>Name</b>        | <b>Description</b>   | <b>Mandatory/<br/>Optional</b> |
|--------------------|--|--------------------------------|
| ID                 | The ID of the reservation.   | Mandatory                      |
| Bandwidth          | The resulting bandwidth. This is within the bandwidth range given in the RESV_REQ. A range is specified using the mandatory bandwidth parameter and the optional minimal acceptable bandwidth parameter. | Optional                       |
| Source mask length | The resulting source mask length. The BM may have generalized the source prefix.   | Optional                       |

|                                     |  |          |
|-------------------------------------|--|----------|
| Destination mask length             | The resulting destination mask length. The BM may have generalized the destination prefix. | Optional |
| Additional source mask lengths      | The BM may have generalized the source prefixes.   | Optional |
| Additional destination mask lengths | The BM may have generalized the destination prefixes.                                      | Optional |
| Ingress address                     | Report on selected ingress address.  | Optional |
| Egress address                      | Report on selected egress address.   | Optional |

#### **4.11 Commit Reservation Request (COMMIT\_REQ)**

This is the second phase in the two-phase model where the reservation is committed and implemented in the edge routers using IF-3. The ID refers to the previously reserved resources (using the RESV\_REQ) and further information is included to identify the flow.

| <b>Name</b>                 | <b>Mandatory/Optional</b> |
|-----------------------------|---------------------------|
| ID of requested reservation | Mandatory                 |
| Source port                 | Mandatory                 |
| Destination port            | Mandatory                 |
| Protocol                    | Mandatory                 |
| Source port low             | Optional                  |
| Destination port low        | Optional                  |
| DSCP                        | Optional                  |
| NAT in from address         | Optional                  |
| NAT in from port            | Optional                  |
| NAT in to address           | Optional                  |
| NAT in to port              | Optional                  |
| NAT out from address        | Optional                  |
| NAT out from port           | Optional                  |
| NAT out to address          | Optional                  |
| NAT out to port             | Optional                  |
| NAT traversal               | Optional                  |

Ports were optional at “resv” and mandatory at “commit” to avoid committing without specifying ports. Ports can be specified as a range for core capacity reservations.

#### **4.12 Commit Reservation Reply (COMMIT\_REPLY)**

If the COMMIT\_REQ succeeded the ID of each reservation committed is returned. If the commit operation fails an error message is returned. However, the resources may still be reserved and therefore a release reservation must be sent.

| Name | Description                | Mandatory/<br>Optional |
|------|----------------------------|------------------------|
| ID   | The ID of the reservation. | Mandatory              |

#### **4.13 Reserve-and-Commit Request (RESVCM\_REQ)**

This message is used for single-phase reservation where both the resource specification and the filter specification are included and if the admission control succeeds the reservation is committed directly without a specific commit request.

| Name                            | Mandatory/<br>Optional |
|---------------------------------|------------------------|
| Bandwidth                       | Mandatory              |
| Network service class           | Mandatory              |
| Priority                        | Mandatory              |
| Source address                  | Mandatory              |
| Source mask length              | Mandatory              |
| Destination address             | Mandatory              |
| Destination mask length         | Mandatory              |
| Source port                     | Mandatory              |
| Destination port                | Mandatory              |
| Protocol                        | Mandatory              |
| VPN ID                          | Optional               |
| Minimal acceptable bandwidth    | Optional               |
| Partition                       | Optional               |
| Ingress address                 | Optional               |
| Egress address                  | Optional               |
| Egress mode                     | Optional               |
| Additional source prefixes      | Optional               |
| Additional destination prefixes | Optional               |
| Start time                      | Optional               |
| Stop time                       | Optional               |
| Time options                    | Optional               |
| User                            | Optional               |
| Source port low                 | Optional               |
| Destination port low            | Optional               |
| DSCP                            | Optional               |
| NAT in from address             | Optional               |
| NAT in from port                | Optional               |
| NAT in to address               | Optional               |
| NAT in to port                  | Optional               |

|                      |          |
|----------------------|----------|
| NAT out from address | Optional |
| NAT out from port    | Optional |
| NAT out to address   | Optional |
| NAT out to port      | Optional |
| NAT traversal        | Optional |

#### **4.14 Reserve-and-Commit Reply (RESVCM\_REPLY)**

This reply has the same format as the RESV\_REPLY.

#### **4.15 Modify Reservation Request (MODIFY\_REQ)**

This message is used by the client to request a modification of an existing reservation. Only modification of the bandwidth amount is mandatory for the BM to support. The operation to modify bandwidth is often used on pre-allocated aggregate resources (e.g., for multiple BMs in a domain and BM peering).

| <b>Name</b>                     | <b>Mandatory/<br/>Optional</b> |
|---------------------------------|--------------------------------|
| ID                              | Mandatory                      |
| Bandwidth                       | Mandatory                      |
| Network service class           | Optional                       |
| Priority                        | Optional                       |
| Source address                  | Optional                       |
| Source mask length              | Optional                       |
| Destination address             | Optional                       |
| Destination mask length         | Optional                       |
| Source port                     | Optional                       |
| Destination port                | Optional                       |
| Protocol                        | Optional                       |
| VPN ID                          | Optional                       |
| Minimal acceptable bandwidth    | Optional                       |
| Partition                       | Optional                       |
| Ingress address                 | Optional                       |
| Egress address                  | Optional                       |
| Egress mode                     | Optional                       |
| Additional source prefixes      | Optional                       |
| Additional destination prefixes | Optional                       |
| Start time                      | Optional                       |
| Stop time                       | Optional                       |
| Time options                    | Optional                       |
| User                            | Optional                       |
| Source port low                 | Optional                       |

|                      |          |
|----------------------|----------|
| Destination port low | Optional |
| DSCP                 | Optional |
| NAT in from address  | Optional |
| NAT in from port     | Optional |
| NAT in to address    | Optional |
| NAT in to port       | Optional |
| NAT out from address | Optional |
| NAT out from port    | Optional |
| NAT out to address   | Optional |
| NAT out to port      | Optional |
| NAT traversal        | Optional |

#### **4.16 Modify Reservation Reply (MODIFY\_REPLY)**

This reply is has exactly the same format as the RESV\_REPLY.

#### **4.17 Release Reservation Request (RELEASE\_REQ)**

This message must be sent to release an open-ended reservation. The resources will be de-allocated and configuration removed from the edge routers using IF-3.

| Name | Description                | Mandatory/<br>Optional |
|------|----------------------------|------------------------|
| ID   | The ID of the reservation. | Mandatory              |

#### **4.18 Release Reservation Reply (RELEASE\_REPLY)**

The reply includes the ID of each reservation released.

| Name                  | Description                          | Mandatory/<br>Optional |
|-----------------------|--------------------------------------|------------------------|
| ID                    | The ID of the reservation.           | Mandatory              |
| Duration              | May be used for accounting purposes. | Optional               |
| Bandwidth             | May be used for accounting purposes. | Optional               |
| Network service class | May be used for accounting purposes. | Optional               |
| User                  | May be used for accounting purposes. | Optional               |

#### **4.19 Reservation Termination Callback (TERM\_CB)**

The BM will send this message unsolicited to the client when a reservation must be pre-empted for any reason. An ID is normally provided. If ID is not provided, auditing is needed to determine which reservations are still active.

| Name          | Description  | Mandatory/<br>Optional |
|---------------|--|------------------------|
| ID            | The ID of the reservation.   | Optional               |
| Reason code   | Predefined code specifying the reason why reservation was terminated. <ol style="list-style-type: none"> <li>1. Pre-empted by high priority call (ID supplied)</li> <li>2. Network failure (ID may or may not be returned)</li> <li>3. BM failure (ID may or may not be returned)</li> </ol> More codes are to be defined. | Mandatory              |
| Reason string | A human-readable textual description of the reason why the reservation was terminated  | Mandatory              |

#### 4.20 Reservation Get Request (GET\_REQ)

The GET\_REQ is used to get known reservations. If no parameters are given, all known reservations are returned in the GET\_REPLY. If the ID is given, only the reservation with matching ID is returned. If the User parameter is given, only the reservations with matching User field are returned. Supplying both ID and User in the same message is not allowed.

If there is no match to any reservation, an error message is returned.

This command is typically used when recovering state.

| Name                        | Mandatory/<br>Optional |
|-----------------------------|------------------------|
| ID of requested reservation | Optional               |
| User                        | Optional               |

#### 4.21 Reservation Get Reply (GET\_REPLY)

If the GET\_REPLY contains all matching reservations there is one parameter set per reservation.

| Name                  | Mandatory/<br>Optional |
|-----------------------|------------------------|
| ID                    | Mandatory              |
| Bandwidth             | Mandatory              |
| Network service class | Mandatory              |
| Priority              | Mandatory              |
| Source address        | Mandatory              |
| Source mask length    | Mandatory              |
| Destination address   | Mandatory              |

MSF-IA-NRCP.001-FINAL  
Implementation agreement for Network Resource Control Protocol (NRCP)

|                                 |           |
|---------------------------------|-----------|
| Destination mask length         | Mandatory |
| Source port                     | Mandatory |
| Destination port                | Mandatory |
| Protocol                        | Mandatory |
| VPN ID                          | Optional  |
| Minimal acceptable bandwidth    | Optional  |
| Partition                       | Optional  |
| Ingress address                 | Optional  |
| Egress address                  | Optional  |
| Egress mode                     | Optional  |
| Additional source prefixes      | Optional  |
| Additional destination prefixes | Optional  |
| Start time                      | Optional  |
| Stop time                       | Optional  |
| Time options                    | Optional  |
| User                            | Optional  |
| Source port low                 | Optional  |
| Destination port low            | Optional  |
| DSCP                            | Optional  |
| NAT in from address             | Optional  |
| NAT in from port                | Optional  |
| NAT in to address               | Optional  |
| NAT in to port                  | Optional  |
| NAT out from address            | Optional  |
| NAT out from port               | Optional  |
| NAT out to address              | Optional  |
| NAT out to port                 | Optional  |
| NAT traversal                   | Optional  |

## 5. Appendix A – NRCP Java API

---

---

### 5.1 Interface *BMCallback*

---

public interface **BMCallback**

Call-back for reservation termination notifications.

---

#### Method Summary

|      |  |
|------|--|
| void | <a href="#">reservationTermination</a> (int code, java.lang.String err, BMReservation resv)<br>Call-back for reservation termination |
|------|--|

---

#### Method Detail

##### 5.1.1 reservationTermination

public void **reservationTermination**(int code,  
  java.lang.String err,  
  BMReservation resv)

Call-back for reservation termination

**Parameters:**

`code` - Error code

`err` - Error string

`resv` - A copy of the terminated reservation object, might be "null".

---

### 5.2 Class *BMException*

java.lang.Object

|

+--java.lang.Throwable

|

+--java.lang.Exception

|

+--**BMException**

**All Implemented Interfaces:**

java.io.Serializable

---

public class **BMException**

extends java.lang.Exception

Bandwidth Manager general exception class.

**See Also:**

[Serialized Form](#)

---

## Constructor Summary

[BMException](#)(int code, java.lang.String err)  
Bandwidth Manager general exception.

[BMException](#)(int code, java.lang.String err, java.lang.Throwable cause)  
Bandwidth Manager general exception with cause.

## Methods inherited from class java.lang.Throwable

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

## Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

## Constructor Detail

### 5.2.1 BMException

```
public BMException(int code,  
                   java.lang.String err)  
                   Bandwidth Manager general exception.
```

**Parameters:**

code - Error code  
err - Error string

### 5.2.2 BMException

```
public BMException(int code,  
                   java.lang.String err,  
                   java.lang.Throwable cause)  
                   Bandwidth Manager general exception with cause.
```

**Parameters:**

code - Error code  
err - Error string

cause - Exception causing this exception

---

### 5.3 Class *BMFatalException*

java.lang.Object

```
|
|--java.lang.Throwable
|
|--java.lang.Exception
|
|--BMFatalException
```

#### All Implemented Interfaces:

java.io.Serializable

---

public class **BMFatalException**

extends java.lang.Exception

Bandwidth Manager fatal exception class. A fatal exception causes the session to close.

#### See Also:

[Serialized Form](#)

---

### Constructor Summary

[BMFatalException](#)(int code, java.lang.String err)  
Bandwidth Manager fatal exception.

[BMFatalException](#)(int code, java.lang.String err, java.lang.Throwable cause)  
Bandwidth Manager fatal exception with cause

### Methods inherited from class java.lang.Throwable

fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

### Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

### Constructor Detail

#### 5.3.1 **BMFatalException**

public **BMFatalException**(int code,

```
java.lang.String err)
```

Bandwidth Manager fatal exception.

**Parameters:**

code - Error code

err - Error string

---

### 5.3.2 BMFatalException

```
public BMFatalException(int code,
    java.lang.String err,
    java.lang.Throwable cause)
```

Bandwidth Manager fatal exception with cause

**Parameters:**

code - Error code

err - Error string

cause - Exception causing this exception

---

## 5.4 Class BMReservation

```
java.lang.Object
```

```
|
```

```
+--BMReservation
```

---

```
public class BMReservation
```

```
extends java.lang.Object
```

Bandwidth Manager Resource handling API. This is a pseudo-coded Java API. Only the main methods are shown. A fully working interface requires some additional access methods.

---

### Constructor Summary

```
BMReservation(BMSession session, int bw, int service, int priority,
    java.lang.String srcAddr, java.lang.String dstAddr)
```

Make a basic open-ended resource reservation, reserve only.

```
BMReservation(boolean commit, BMSession session, int bw, int minBw,
    int service, int priority, java.lang.String srcAddr, int srcMask,
    java.lang.String dstAddr, int dstMask, int VPNid, int partition,
    java.lang.String ingressAddr, java.lang.String egressAddr,
    boolean egressRequired, java.util.Calendar startTime,
    java.util.Calendar stopTime, int timeOption, java.lang.String user,
    int srcPortLow, int srcPortHigh, int dstPortLow, int dstPortHigh, int proto,
    int natTraversalMode, java.lang.String natInFromAddr, int natInFromPort,
    java.lang.String natInToAddr, int natInToPort,
    java.lang.String natOutFromAddr, int natOutFromPort,
    java.lang.String natOutToAddr, int natOutToPort)
```

Make a fully specified resource reservation.

## Implementation agreement for Network Resource Control Protocol (NRCP)

**BMRreservation**(boolean commit, BMSession session, int bw, int service, int priority, java.lang.String srcAddr, java.lang.String dstAddr, int srcPort, int dstPort, int proto)  
 Make a basic open-ended resource reservation.

**Method Summary**

|                  |  |
|------------------|--|
| void             | <a href="#"><u>commit</u></a> (int srcPort, int dstPort, int proto)<br>Commit reservation, with single ports.  |
| void             | <a href="#"><u>commit</u></a> (int srcPortLow, int srcPortHigh, int dstPortLow, int dstPortHigh, int proto)<br>Commit reservation, with port ranges.   |
| java.lang.String | <a href="#"><u>egress</u></a> ()<br>Return the egress address  |
| int              | <a href="#"><u>id</u></a> ()<br>Return the reservation id  |
| java.lang.String | <a href="#"><u>ingress</u></a> ()<br>Return the ingress address  |
| void             | <a href="#"><u>modify</u></a> (int bw)<br>Modify bandwidth   |
| void             | <a href="#"><u>modify</u></a> (int bw, int minBw, int service, int priority, java.lang.String srcAddr, int srcMask, java.lang.String dstAddr, int dstMask, int srcPortLow, int srcPortHigh, int dstPortLow, int dstPortHigh, int proto, int VPNid, int partition, java.lang.String ingressAddr, java.lang.String egressAddr, boolean egressRequired, java.util.Calendar startTime, java.util.Calendar stopTime, int timeOption, java.lang.String user)<br>Modify one or more reservation attributes. |
| void             | <a href="#"><u>stop</u></a> ()<br>Stop reservation.  |

**Methods inherited from class java.lang.Object**

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

**Constructor Detail****5.4.1 BMRreservation**

```
public BMRreservation(BMSession session,
    int bw,
    int service,
```

```

    int priority,
    java.lang.String srcAddr,
    java.lang.String dstAddr)

```

throws [BMException](#),  
[BMFatalException](#)

Make a basic open-ended resource reservation, reserve only.

**Parameters:**

```

    session - Reference to a BM session
    bw - Requested bandwidth (kbps)
    service - Service
    priority - Reservation priority
    srcAddr - Source address
    dstAddr - Destination address

```

---

#### 5.4.2 BMReservation

public **BMReservation**(boolean commit,

```

    BMSession session,
    int bw,
    int service,
    int priority,
    java.lang.String srcAddr,
    java.lang.String dstAddr,
    int srcPort,
    int dstPort,
    int proto)

```

throws [BMException](#),  
[BMFatalException](#)

Make a basic open-ended resource reservation.

**Parameters:**

```

    commit - True if this is "reserve and commit", false if "reserve only"
    session - Reference to a BM session
    bw - Requested bandwidth (kbps)
    service - Service
    priority - Reservation priority
    srcAddr - Source address
    dstAddr - Destination address
    srcPort - Source port number
    dstPort - Destination port number
    proto - Protocol number

```

---

### 5.4.3 BMRreservation

```
public BMRreservation(boolean commit,  
    BMSession session,  
    int bw,  
    int minBw,  
    int service,  
    int priority,  
    java.lang.String srcAddr,  
    int srcMask,  
    java.lang.String dstAddr,  
    int dstMask,  
    int VPNid,  
    int partition,  
    java.lang.String ingressAddr,  
    java.lang.String egressAddr,  
    boolean egressRequired,  
    java.util.Calendar startTime,  
    java.util.Calendar stopTime,  
    int timeOption,  
    java.lang.String user,  
    int srcPortLow,  
    int srcPortHigh,  
    int dstPortLow,  
    int dstPortHigh,  
    int proto,  
    int natTraversalMode,  
    java.lang.String natInFromAddr,  
    int natInFromPort,  
    java.lang.String natInToAddr,  
    int natInToPort,  
    java.lang.String natOutFromAddr,  
    int natOutFromPort,  
    java.lang.String natOutToAddr,  
    int natOutToPort)  
throws BMException,  
    BMFatalException
```

Make a fully specified resource reservation. Some parameters are optional. These parameters should be set to zero, null or false (depending on type) when not used.

#### Parameters:

`commit` - True if this is "reserve and commit", false if "reserve only"  
`session` - Reference to a BM session  
`bw` - Requested bandwidth (kbps)

## Implementation agreement for Network Resource Control Protocol (NRCP)

minBw - Minimal acceptable bandwidth (\*Optional)  
 service - Service  
 priority - Reservation priority  
 srcAddr - Source address  
 srcMask - Source address mask length (\*Optional)  
 dstAddr - Destination address  
 dstMask - Destination address mask length (\*Optional)  
 VPNid - MPLS VPN id (\*Optional)  
 partition - Resource partition (\*Optional)  
 ingressAddr - Ingress address (domain entry point) (\*Optional)  
 egressAddr - Egress address (domain exit point) (\*Optional)  
 egressRequired - Egress is required. Egress treated as a hint if false. (\*Optional)  
 startTime - Start time of reservation (\*Optional)  
 stopTime - Stop time of reservation (\*Optional)  
 timeOption - (Open-ended, Relative, Absolute, Periodic) (\*Optional)  
 user - External user name (\*Optional)  
 srcPortLow - Source port number, lower bound (\*Optional)  
 srcPortHigh - Source port number, upper bound  
 dstPortLow - Destination port number, lower bound (\*Optional)  
 dstPortHigh - Destination port number, upper bound  
 proto - Protocol number  
 natTraversalMode - if specified NAT traversal is required but the NAT mappings are unknown (\*Optional). 1 = NAT traversal at both ingress and egress. 2 = NAT traversal at ingress only. 3 = NAT traversal at egress only.  
 natInFromAddr - source address arriving to ingress (\*Optional)  
 natInFromPort - source port number arriving to ingress (\*Optional)  
 natInToAddr - destination address arriving to ingress (\*Optional)  
 natInToPort - destination port number arriving to ingress (\*Optional)  
 natOutFromAddr - source address leaving from egress (\*Optional)  
 natOutFromPort - source port number leaving from egress (\*Optional)  
 natOutToAddr - destination address leaving from egress (\*Optional)

## Method Detail

### 5.4.4 commit

```

public void commit(int srcPort,
                   int dstPort,
                   int proto)
    throws BMException,
           BMFatalException
  
```

Commit reservation, with single ports. Exception raised if this is a "reserve and commit" type reservation.

**Parameters:**

srcPort - Source port number  
dstPort - Destination port number  
proto - Protocol number  
[BMException](#)  
[BMFatalException](#)

---

**5.4.5 commit**

```
public void commit(int srcPortLow,  
    int srcPortHigh,  
    int dstPortLow,  
    int dstPortHigh,  
    int proto)
```

throws [BMException](#),  
[BMFatalException](#)

Commit reservation, with port ranges. Exception raised if this is a "reserve and commit" type reservation.

**Parameters:**

srcPortLow - Source port number, lower bound  
srcPortHigh - Source port number, upper bound  
dstPortLow - Destination port number, lower bound  
dstPortHigh - Destination port number, upper bound  
proto - Protocol number  
[BMException](#)  
[BMFatalException](#)

---

**5.4.6 modify**

```
public void modify(int bw)
```

throws [BMException](#),  
[BMFatalException](#)

Modify bandwidth

**Parameters:**

bw - Requested bandwidth (kbps)  
[BMException](#)  
[BMFatalException](#)

---

#### 5.4.7 modify

```
public void modify(int bw,  
    int minBw,  
    int service,  
    int priority,  
    java.lang.String srcAddr,  
    int srcMask,  
    java.lang.String dstAddr,  
    int dstMask,  
    int srcPortLow,  
    int srcPortHigh,  
    int dstPortLow,  
    int dstPortHigh,  
    int proto,  
    int VPNid,  
    int partition,  
    java.lang.String ingressAddr,  
    java.lang.String egressAddr,  
    boolean egressRequired,  
    java.util.Calendar startTime,  
    java.util.Calendar stopTime,  
    int timeOption,  
    java.lang.String user)
```

throws [BMException](#),  
[BMFatalException](#)

Modify one or more reservation attributes. Attributes with non-zero values will be updated.

##### **Parameters:**

`bw` - Requested bandwidth (kbps)  
`minBw` - Minimal acceptable bandwidth  
`service` - Service  
`priority` - Reservation priority  
`srcAddr` - Source address  
`srcMask` - Source address mask length  
`dstAddr` - Destination address  
`dstMask` - Destination address mask length  
`srcPortLow` - Source port number, lower bound  
`srcPortHigh` - Source port number, upper bound  
`dstPortLow` - Destination port number, lower bound  
`dstPortHigh` - Destination port number, upper bound  
`proto` - Protocol number  
`VPNid` - MPLS VPN id

partition - Resource partition  
ingressAddr - Ingress address (domain entry point)  
egressAddr - Egress address (domain exit point)  
egressRequired - Egress is required. Egress treated as a hint if false.  
startTime - Start time of reservation  
stopTime - Stop time of reservation  
timeOption - (Open-ended, Relative, Absolute, Periodic)  
user - External user name  
[BMException](#)  
[BMFatalException](#)

---

#### 5.4.8 stop

public void **stop**()  
throws [BMException](#),  
    [BMFatalException](#)  
Stop reservation.  
[BMException](#)  
[BMFatalException](#)

---

#### 5.4.9 id

public int **id**()  
Return the reservation id

---

#### 5.4.10 ingress

public java.lang.String **ingress**()  
Return the ingress address

---

#### 5.4.11 egress

public java.lang.String **egress**()  
Return the egress address

---

### 5.5 Class *BMSession*

java.lang.Object  
|  
+--**BMSession**

---

public class **BMSession**

extends java.lang.Object

Bandwidth Manager Session setup. This is a pseudo-coded Java API. Only the main methods are shown. A fully working interface requires some additional access methods (for socket/stream references, etc).

## Constructor Summary

[BMSession](#)(java.lang.String host, int port, java.lang.String usr, java.lang.String pw)  
BM Session setup.

## Method Summary

void [close](#)()  
BM Session termination.

## Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

## Constructor Detail

### 5.5.1 BMSession

```
public BMSession(java.lang.String host,
                 int port,
                 java.lang.String usr,
                 java.lang.String pw)
    throws BMException,
           BMFatalException
    BM Session setup.
```

#### Parameters:

host - Hostname of the BM  
port - Port number on the BM  
usr - User name to be authenticated as  
pw - Password string

## Method Detail

### 5.5.2 close

```
public void close()
    throws BMException
    BM Session termination.
```

[BMException](#)