Messaging for a Dynamic World

Approved: Magnus Björkman	Mobeon Internal No: 8/FD-MAS0001 Uen

Copyright Mobeon AB All rights reserved	Author: MMANY, MMAWI, MANDE Title: FD – Call Manager	Version:D Date: 2008-08-05	1/57

FD – Call Manager

Content

History

Version	Date	Adjustments
A	2006-10-03	First reversion. (MMAWI)
B	2006-12-07	Updated disconnect event for: • inbound call states: Alerting-New Call, Alerting-Accepting and Alerting-Early Media. • outbound call states: Progressing-Calling, Progressing-Proceeding and Progressing-Early Media. (MANDE) Updated join/unjoin handling. The following items were updated: • Table 7, 8, 18, 31, 32, 39, 42 • Figure 7, 8 • Text in sections 3.4.1.3 and 3.4.2.3 Removed the following sections: • 3.4.1.3.4 Transfer state • 3.4.2.3.4 Transfer state (MMANY) Updated administrative states in section 3.1 and statistics in section 3.8. (MMAWI)
C	2007-06-13	CallManager no longer waits a short time after "close(unforced)" before unregistering from SSP; now unregistering is done immediately. Administrative state machine in section Administrative state machine3.1.1 updated (ERMKESE). Updated with: - unsolicited SIP NOTIFY - 100rel - Change in response to SIP OPTIONS (MMANY)
D	2008-08-05	Minor changes. Updated load regulation section. (ekensel)

1 Introduction

This document describes the Call Manager functions in detail. The Call Manager functions are specified in [1]. The Call Manager implements a user agent SIP interface as specified in [2].

The Call Manager component is from now on referred to as **CM**.

1.1 Glossary

1.1.1 External Client

An external client is a client (e.g. a SIP user agent) that communicates with **CM** over SIP.

1.1.2 Dialog/Call

Dialog is a term defined in the SIP standard (see [6]).

A session is a peer-to-peer SIP relationship between two user agents that persists for some time. A session is a collection of participants, and streams of media between them, for the purpose of communication. A session is established using the SIP INVITE message, resulting in one or more SIP dialogs (see [6]). A dialog is communication between two parties. Since multicast conferencing is not supported currently one session consists of one single dialog only.

In this document, a dialog is synonym to a call.

1.1.3 Re-INVITE

A re-INVITE is a SIP INVITE request that addresses an already existing dialog, i.e. this is not an INVITE to setup a new dialog but to change the existing dialog.

The handling of INVITE versus a re-INVITE has been separated in this document.

1.1.4 SIP layers

SIP is structured as a layered protocol, which means that its behavior is described in terms of a set of fairly independent processing stages with only a loose coupling between each stage. The layers are defined in [6] and are illustrated in the figure below.

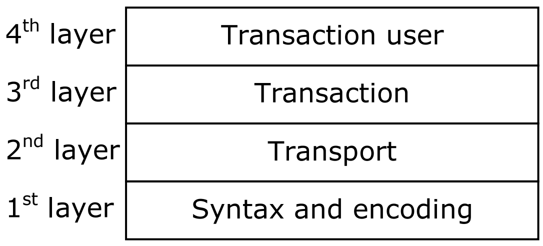


Figure 1 The four SIP layers

The layers are used in the document to illustrate where specific parts of the SIP specification are implemented in the Call Manager.

1.1.5 Transaction

Transaction is a term defined in the SIP standard (see [6]).

A transaction identifies an ongoing message exchange between two parties. On the client side of the message exchange, there is a client transaction and vice versa for the server side of the message exchange.

2 Function Structure

2.1 Overview

The overall structure of **CM** is illustrated in the figure below. The dotted arrow in the figure illustrates normal SIP message flow through **CM**.

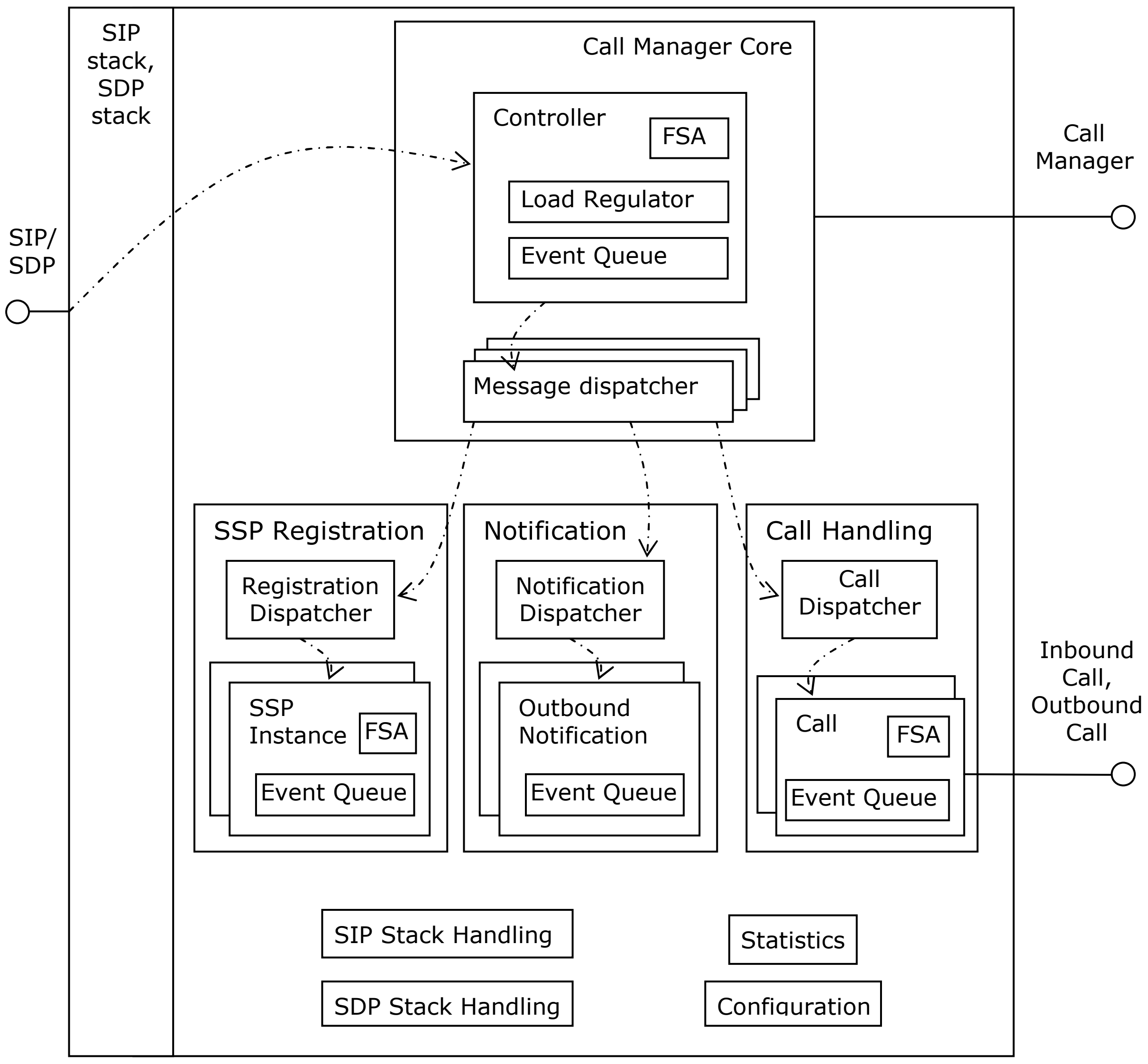


Figure 2 Structure of the Call Manager component

CM also consists of functionality for diagnosing the service enabled by **CM**. This functionality is completely separated from the other **CM** functionality in order to not affect normal **CM** behavior. The structure of the diagnose service functionality is illustrated below.

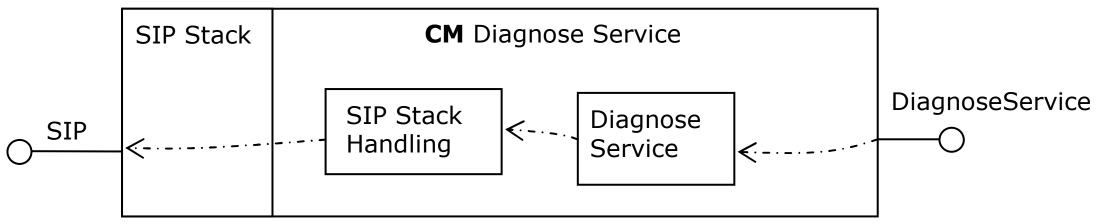


Figure 3 Structure of the diagnose service part of CM

2.1.1 SIP stack

The SIP Stack is responsible for sending/receiving SIP messages over the SIP interface. It parses messages and maintains transaction state.

NIST-SIP is a third-party product (see [3]) that is used as the SIP stack. The NIST-SIP version used is 1.2. The interface towards NIST-SIP is the standardized JAIN SIP API (see [4]).

The SIP stack implements the SIP layers one to three, i.e. up and including the transaction layer. The SIP stack is also able to implement part of the forth layer (Transaction User) since it is possible to configure it to handle retransmissions. When used in **CM**, retransmissions in the SIP stack is activated as described in section 3.7

The SIP stack code is found in the jain_sip vob and the version used by **CM** is built and checked in as jar files in the mas vob.

2.1.2 SDP stack

The SDP Stack is responsible for parsing and generating SDP descriptions.

NIST-SDP is an extension to NIST-SIP (see [3]) that is used as the SDP stack. The interface towards NIST-SDP is implemented in-house and is not standardized. It is however constructed similar to a known example standard called JAIN SDP API (see [5]) in order to be easily exchangeable if JAIN SDP API ever becomes an open standard.

The SDP stack code is found in the jain_sip vob and the version used by **CM** is built and checked in as jar files in the mas vob.

2.1.3 SIP stack handling

The purpose of the SIP stack handling is to concentrate the use of the SIP stack into one part of **CM** in order to more easily be able to exchange the SIP stack to another (not using the JAIN SIP API).

There are parts of the implementation in the SIP stack handling that uses code directly from the SIP stack implementation rather than its interface. The implementation has been done this way in order save time and avoid re-implementing code already supported by the stack (although not through the interface). This kind of implementation should normally be avoided, and must be replaced if the SIP stack ever changes its implementation or if the stack is replaced with another product. It is however limited to a very small part of **CM**.

The SIP stack handling code is found in the package:

- com.mobeon.masp.callmanager.sip

2.1.4 SDP stack handling

The purpose of the SDP stack handling is to concentrate the use of the SDP stack into one part of **CM** in order to more easily be able to exchange the SDP stack to another.

The SDP stack handling code is found in the package:

- com.mobeon.masp.callmanager.sdp

2.1.5 Call Manager Core

The core is responsible for overall management of **CM**. It exports the Call Manager interface that is used to create outbound calls, and join/unjoin two calls. The core is also responsible for keeping track of the administrative state and current load situation in order to redirect or reject calls when the situation requires.

The core mainly consists of a controller and multiple message dispatchers.

The code for the **CM** core is found in the packages:

- com.mobeon.masp.callmanager
- com.mobeon.masp.callmanager.events
- com.mobeon.masp.callmanager.states
- com.mobeon.masp.callmanager.loadregulation

The **CM** core implements the forth SIP layer; "transaction user".

2.1.5.1 Controller

The main purpose if the controller is to keep track of the **CM** status with regards to administrative state and load situation and handle all incoming events as the current status requires.

All incoming events (received either over SIP or using the CallManager interface) are queued in the controllers' event queue. From the event queue, one event at a time is handled in the order they arrived. The controller keeps a state machine for the administrative state (see section 3.1) and has a load regulator that keeps a state machine for the current load situation (see section 3.2).

An event is mainly handled either by sending it directly to the corresponding call or SSP instance (if such exists) using message dispatchers or by injecting it to the administrative state FSA (to determine how it should be handled).

The controller is responsible for creating new inbound/outbound calls and outbound notifications when the current **CM** status allows.

2.1.5.2 Message dispatcher

The message dispatchers are used to dispatch incoming SIP messages. Three message dispatchers exist:

- SIP request dispatcher

- SIP response dispatcher
- SIP timeout dispatcher

Depending on the SIP procedure (INVITE, REGISTER or other) the message is either dispatched to a call (using the call dispatcher), an SSP instance (using the registration dispatcher) or handled out-of-dialog. An out-of-dialog INVITE will result in a new call being created by the **CM** controller if the current status allows.

2.1.6 SSP registration

The SSP registration part of **CM** is responsible for registering to each configured SSP. This part mainly consists of a registration dispatcher and one SSP instance for each configured SSP.

The code for the SSP registration is found in the package:

- com.mobeon.masp.callmanager.registration

2.1.6.1 Registration dispatcher

The registration dispatcher is responsible for dispatching an incoming SIP message to the corresponding registration procedure for a specific SSP instance. How the dispatching is done is further described in section 3.3.2.

2.1.6.2 SSP instance

The SSP instance is responsible for the registration procedure for one specific SSP.

The SSP instance consists of an event queue and a state machine containing the current registration state. Incoming events are queued in the event queue and then handled one at a time in the order they arrived. An incoming event could either be a received SIP message belonging to a REGISTER procedure, or a request to initiate register/unregister coming from the **CM** controller, see usage in section 3.1. When handling an event it is injected to the state machine and the resulting action depends on the state as described in section 3.3.

2.1.7 Notification

The notification part of **CM** is responsible for sending out unsolicited SIP NOTIFY requests when requested over the CallManager interface. This part mainly consists of a notification dispatcher and one Outbound Notification instance for each ongoing notification.

2.1.7.1 Notification dispatcher

The notification dispatcher is responsible for dispatching an incoming SIP message to the corresponding ongoing outbound notification. How the dispatching is done is further described in section 3.4.1.

2.1.7.2 Outbound notification

Each requested unsolicited NOTIFY is represented by an outbound notification.

The outbound notification is responsible for sending the SIP NOTIFY request. The outbound notification also consists of an event queue where incoming events are queued and then handled one at a time in the order they arrived. An incoming event could either be a response to the SIP NOTIFY procedure or an indication that a timeout occurred when sending the NOTIFY request.

2.1.8 Call handling

The call handling part of **CM** is responsible for call management. This part mainly consists of a call dispatcher and one inbound or outbound call for each active call.

The code for the call handling is found in the package:

- com.mobeon.masp.callmanager.callhandling

2.1.8.1 Call dispatcher

The call dispatcher is responsible for dispatching an incoming SIP message to the corresponding call. How the dispatching is done is further described in section 3.5.3.

2.1.8.2 Call

A call represents one active SIP dialog. A call can be one of two types; inbound or outbound. The call provides the InboundCall and OutboundCall interfaces.

The call consists of an event queue and a state machine containing the current call state. Incoming events are queued in the event queue and then handled one at a time in the order they arrived. An incoming event could either be a SIP INVITE, a SIP message received within dialog, a close request from the CM controller or a request from the provided interface.

When handling an event it is injected to the state machine and the resulting action depends on the state as described in section 3.5.1 for an inbound call and in section 3.5.2 for an outbound call.

The call is also a container for all call specific information, e.g. the calling party of a call.

2.1.9 Statistics

CM collects statistics as specified in [1]. The details on how statistics are calculated can be found in section 3.7.

The code for the statistics is found in the package:

- com.mobeon.masp.callmanager.statistics

2.1.10 Configuration

The configuration of **CM** is handled by a configuration reader and a configuration container. The details on how configuration is handled can be found in section 3.10.

The code for the configuration handling is found in the package:

- com.mobeon.masp.callmanager.configuration

3 Function Behavior

3.1 O&M administration

CM implements the ServiceEnabler interface (as specified in [1]) for O&M administration. A state machine is used to keep track of the administrative state.

3.1.1 Administrative state machine

This section describes the state machine for the administrative state of **CM**.

3.1.1.1 States

The table below lists the states that exist for the administrative state machine.

Table 1 States in administrative state machine

State	Description
Closed	In the *Closed* state a previous forced close or unforced close request by O&M has completed and all calls have terminated. No new calls are accepted by **CM**.
ClosedForcing	In the *ClosingForced* state **CM** has been requested to force close by O&M. For **CM** this implies immediately terminating all active calls and not accepting any new calls. As soon as all calls have terminated a transition is made to the *Closed* state.
ClosingUnforced Rejecting	In the *ClosingUnforcedRejecting* state **CM** has been requested to unforce close by O&M. For **CM** this implies unregistering from all SSPs and not accepting new calls. Currently active calls to keep running until terminated by end-user.
Opened	In the *Opened* state **CM** are able to accept traffic from an O&M point of view. However, the load situation may still affect the ability to accept new calls as described in section 3.2.

3.1.1.2 Events

The table below lists the events that exist for the administrative state machine.

Table 2 Events in administrative state machine

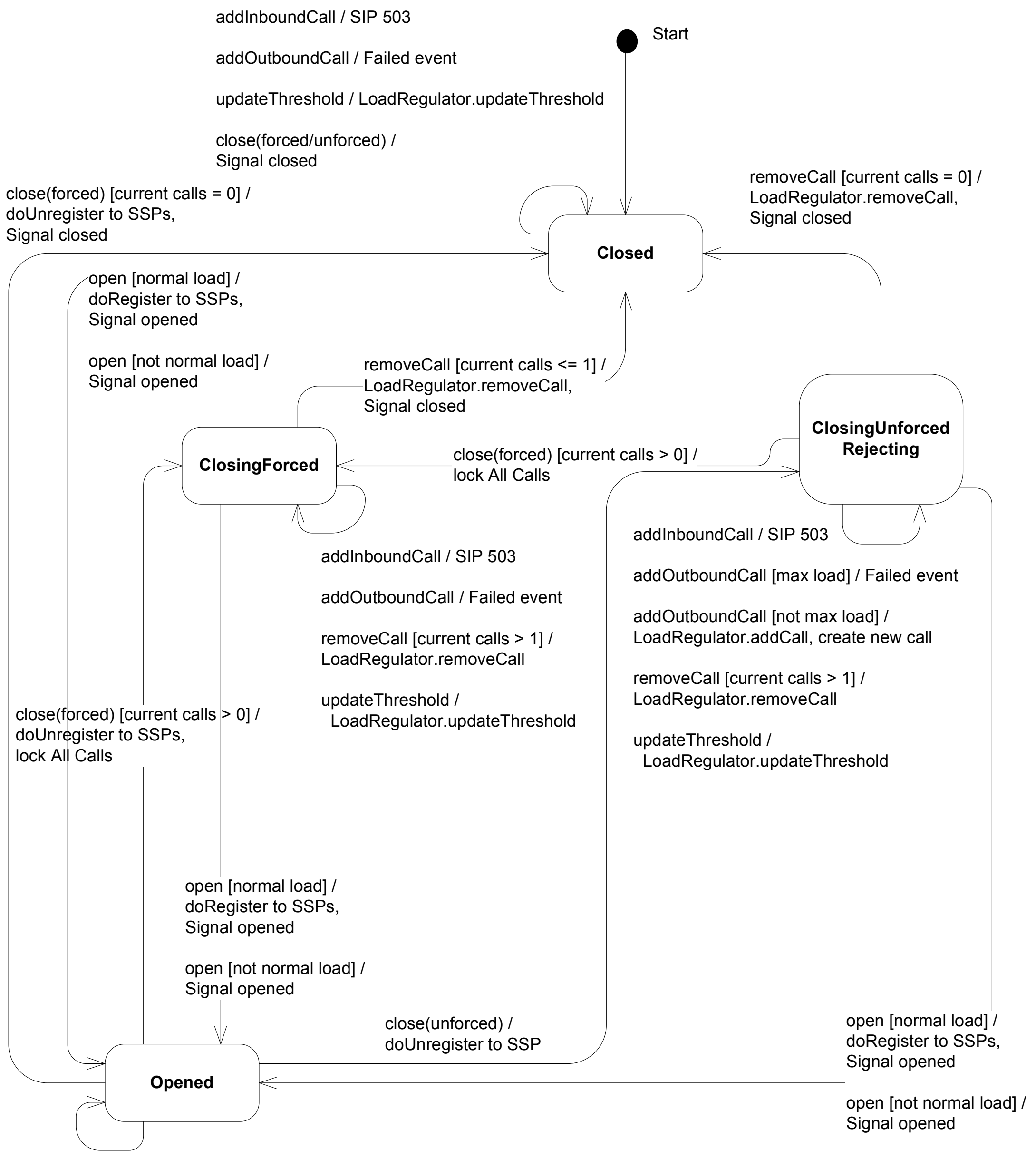
Event	Description
closeForced	O&M has requested **CM** to force close.
open	O&M has requested **CM** to open.
closeUnforced	O&M has requested **CM** to unforce close.
addInboundCall	A session creating SIP INVITE has been received by **CM**.

Event	Description
checkCallAction	A SIP OPTIONS request has been received by **CM**. The response to a SIP OPTIONS request should (according to specification) reflect how a SIP INVITE request would be handled. The purpose of this event is therefore to check what would happen if an inbound call where to be added but without actually performing the changes done for the addInboundCall event. According to specifications, a SIP OPTIONS request should be responded in the same way as for a SIP INVITE. However, **CM** responds to SIP INVITE and OPTIONS requests in the following way: • A SIP INVITE will be accepted until max load has been reached. • A SIP OPTIONS request will be rejected as soon as high water mark has been reached. The above is not entirely according to specification since the specification states that the response to a SIP OPTIONS request should be the same had it been a SIP INVITE request. However, the reason for not following the specification in this situation is to be able to send out an early warning that the system is experiencing high load.
addOutboundCall	A request to create a new outbound call has been received by **CM** over the CallManager interface.
removeCall	A call has completed execution due to for example hang up or failure and requests to be removed. This event is the only one not initiated by events over external interfaces such as SIP or CallManager. This event is needed by the administrative FSA to be able to control load regulation and SSP registration.
updateThreshold	O&M has requested to update the threshold used during load regulation.

3.1.1.3 FSA

The complete administrative state machine is illustrated in the figure below.

All events not included in the figure results in no action. Therefore checkCallAction is not included in the figure since it always results in no action but merely indicates whether a new inbound call would be accepted or not.


addInboundCall / SIP 503
addOutboundCall / Failed event
updateThreshold / LoadRegulator.updateThreshold
close(forced/unforced) / Signal closed
Start
close(forced) [current calls = 0] / doUnregister to SSPs, Signal closed
removeCall [current calls = 0] / LoadRegulator.removeCall, Signal closed
Closed
open [normal load] / doRegister to SSPs, Signal opened
open [not normal load] / Signal opened
removeCall [current calls <= 1] / LoadRegulator.removeCall, Signal closed
ClosingUnforced Rejecting
ClosingForced
close(forced) [current calls > 0] / lock All Calls
addInboundCall / SIP 503
addOutboundCall / Failed event
removeCall [current calls > 1] / LoadRegulator.removeCall
updateThreshold / LoadRegulator.updateThreshold
addInboundCall / SIP 503
addOutboundCall [max load] / Failed event
addOutboundCall [not max load] / LoadRegulator.addCall, create new call
removeCall [current calls > 1] / LoadRegulator.removeCall
updateThreshold / LoadRegulator.updateThreshold
close(forced) [current calls > 0] / doUnregister to SSPs, lock All Calls
open [normal load] / doRegister to SSPs, Signal opened
open [not normal load] / Signal opened
close(unforced) / doUnregister to SSP
open [normal load] / doRegister to SSPs, Signal opened
open [not normal load] / Signal opened
Opened

addInboundCall / LoadRegulator.addCall
If current load requires traffic stop: doUnregister to SSPs
If current load requires traffic redirection: SIP 302 (or SIP 503)
Otherwise: create new call

addOutboundCall [max load] / Failed event

addOutboundCall [not max load] / LoadRegulator.addCall, create new call
If current load requires traffic stop: doUnregister to SSPs

removeCall / LoadRegulator.removeCall
If current load requires traffic start: doRegister to SSPs

updateThreshold / LoadRegulator.updateThreshold,
If current load requires traffic start: doRegister to SSPs
If current load requires traffic stop: doUnregister to SSPs

open /
Signal opened

Figure 4 Administrative state machine

3.2 Load regulation

CM uses a state machine to keep track of the current load situation.

3.2.1 Load regulation state machine

This section describes the state machine for the **CM** load regulation.

3.2.1.1 States

The table below lists the states that exist for the load regulation state machine.

Table 3 States in load regulation state machine

State	Description
High load	The amount of current calls has reached high water mark (HWM).
Normal load	The amount of current calls is below high water mark (HWM).
Max load	The amount of current calls has reached the max amount of calls.

3.2.1.2 Events

The table below lists the events that exist for the load regulation state machine.

Table 4 Events in load regulation state machine

Event	Description
addCall	A new inbound or outbound call is added.
removeCall	An existing call is removed.
updateThreshold	The thresholds for low water mark, high water mark and maximum calls are updated.

3.2.1.3 FSA

The complete load regulation state machine is illustrated in the figure below.

All events not included in the figure results in no action.

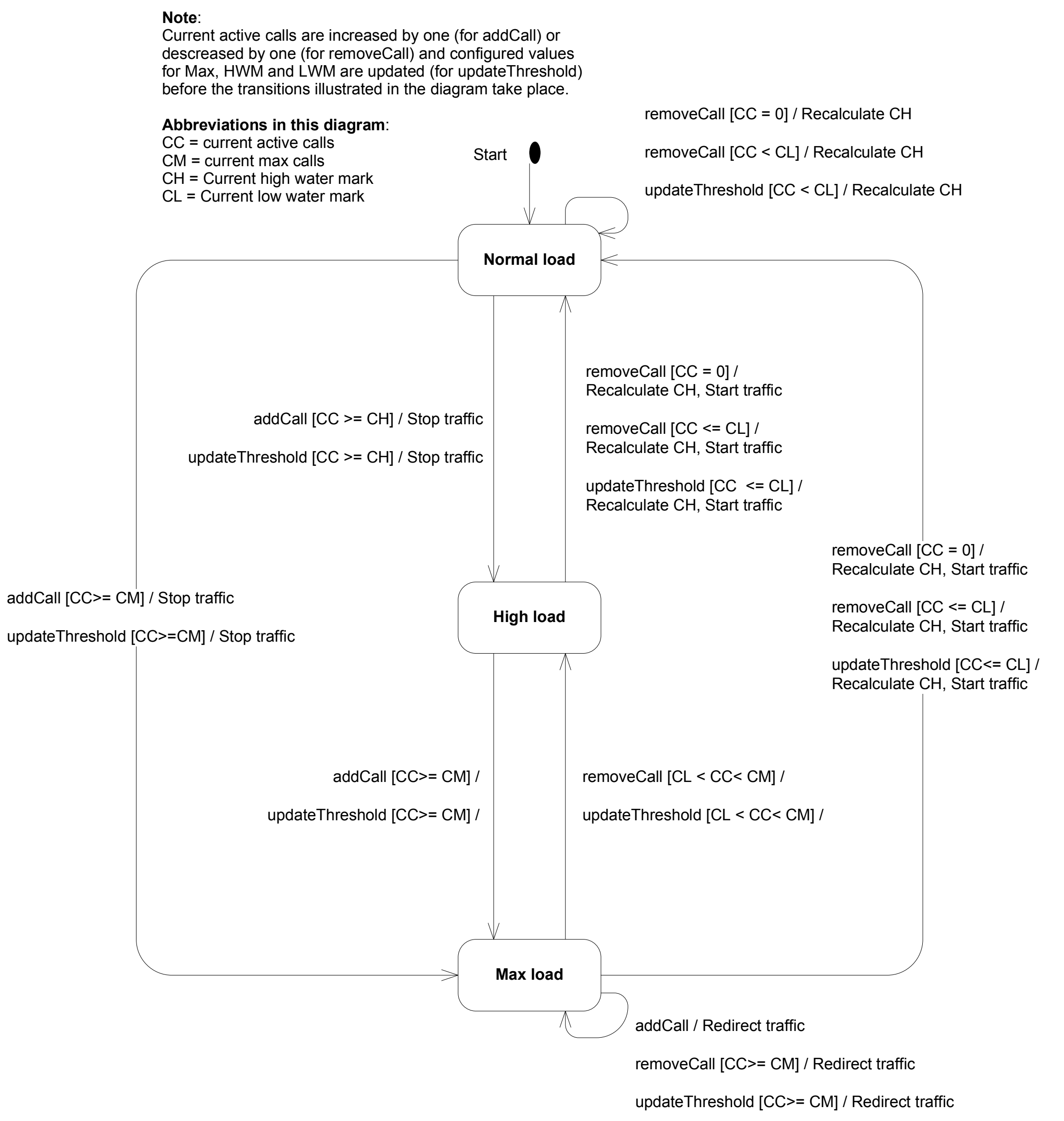

Note:
Current active calls are increased by one (for addCall) or descreased by one (for removeCall) and configured values for Max, HWM and LWM are updated (for updateThreshold) before the transitions illustrated in the diagram take place.
Abbreviations in this diagram:
CC = current active calls
CM = current max calls
CH = Current high water mark
CL = Current low water mark
Start
removeCall [CC = 0] / Recalculate CH
removeCall [CC < CL] / Recalculate CH
updateThreshold [CC < CL] / Recalculate CH
Normal load
addCall [CC >= CH] / Stop traffic
updateThreshold [CC >= CH] / Stop traffic
removeCall [CC = 0] / Recalculate CH, Start traffic
removeCall [CC <= CL] / Recalculate CH, Start traffic
updateThreshold [CC <= CL] / Recalculate CH, Start traffic
removeCall [CC = 0] / Recalculate CH, Start traffic
removeCall [CC <= CL] / Recalculate CH, Start traffic
updateThreshold [CC<= CL] / Recalculate CH, Start traffic
addCall [CC>= CM] / Stop traffic
updateThreshold [CC>=CM] / Stop traffic
High load
addCall [CC>= CM] /
updateThreshold [CC>= CM] /
removeCall [CL < CC< CM] /
updateThreshold [CL < CC< CM] /
Max load
addCall / Redirect traffic
removeCall [CC>= CM] / Redirect traffic
updateThreshold [CC>= CM] / Redirect traffic

Figure 5 Load regulation state machine

3.2.2 Load regulation during startup

During startup, MAS cannot take the maximum amount of load immediately. This is because of the JIT compiler of the JVM. A slow ramp up of the load is required.

3.2.2.1 Algorithm

MAS has a set of configured values: *threshold*, *hwm* and *lwm*. In addition to these, MAS also has a *currentThreshold*, *currentHwm* and *currentLwm* which are the limits currently used.

The current-limits will move towards the configured values by a ramp factor *n/m* channels when the high load state ceases.

threshold	is the absolute maximum numbers of channels the MAS will accept.
currentThreshold	is the current maximum number of channels the MAS will accept. This value will move from a low initial value towards *threshold*.
n/m	is the fraction to increase *currentThreshold*. It is calculated as *n* channels per *m* high load state ceases.

Startup:

1. *threshold*, *hwm* and *lwm* are set by OMM according to the configuration.
2. Initial *currentHwm* is given by configuration, typically 1.
3. *currentLwm* is calculated from *currentHwm* - (*hwm* - *lwm*). If the result is below 0, 0 is used.
4. *currentThreshold* is calculated from *currentHwm* + (*threshold* - *hwm*). If the result is below 0, 0 is used.
5. *n* and *m* are given by configuration.

When the number of channels exceeds *currentHwm*:

1. MAS unregisters from the SSP.

When the number of channels exceeds *currentThreshold*:

1. MAS starts to redirect incoming SIP INVITEs to the SSPs.

When the number of channels reaches *currentLwm* (after recently have reached currentHwm):

1. *currentHwm* is increased by *n/m* channels. If the result is higher than *hwm*, *hwm* is used.
2. *currentLwm* is calculated from *currentHwm* - (*hwm* - *lwm*). If the result is below 0, 0 is used.
3. *currentThreshold* is calculated from *currentHwm* + (*threshold* - *hwm*). If the result is below 0, 0 is used.
4. If *currentThreshold* reaches threshold, *currentThreshold* is set to threshold, *currentHwm* is set to *hwm* and *currentLwm* is set to *lwm*.
5. MAS registers in an SSP.

The load regulation ramp up can be disabled by configuring the initial *currentHwm* to 0. The values from OMM are then used as *threshold*, *hwm* and *lwm*.

3.3 SSP registration

CM is configured with a list of SSPs to use for outbound calls. **CM** registers itself in the SSPs to be able to receive inbound calls.

CM is configured with a list of all SSP:s to register with. Therefore, when **CM** sends a SIP REGISTER request the *Max Forwards* header is set to zero in order to make sure that the SSP does not forward the request on to another SSP.

CM uses a state machine for each configured SSP to keep track of the current registration procedure.

3.3.1 Registration state machine

This section describes the state machine for the SSP registration.

3.3.1.1 States

The table below lists the states that exist for the registration state machine.

Table 5 States in registration state machine

State	Description
Unregistered	In the *Unregistered* state **CM** is not registered in the SSP and there is no ongoing registration procedure. No outbound calls are initiated to this SSP.
Unregistering	In the *Unregistering* state **CM** is in an ongoing procedure to unregister from an SSP. No outbound calls are initiated to this SSP.
Registered	In the *Registered* state **CM** is registered in the SSP. Outbound calls can be initiated to this SSP.
Registering	In the *Registering* state **CM** is in an ongoing procedure to register in an SSP. If the previous state was *Registered*, outbound calls can be initiated to this SSP. Otherwise, no outbound calls are initiated to this SSP.

3.3.1.2 Events

The table below lists the events that exist for the registration state machine.

Table 6 Events in registration state machine

Event	Description
doRegister	A registration procedure shall be initiated to the SSP.
doUnregister	An unregistration procedure shall be initiated to the SSP.
SIP OK	A SIP 2xx response has been received for the registration procedure.
SIP Error	A SIP 3xx or 4xx-6xx response has been received for the registration procedure.
SIP Timeout	A SIP timeout has occurred for the registration procedure.
Re-register timer expires	The re-register timer has expired which means that a new registration procedure shall be initiated. This timer is used to refresh the current registration in the SSP.
Backoff timer expires	The backoff timer has expired which means a new registration procedure shall be initiated. This timer is used to wait a while before a new registration if an error occurred in the last registration procedure.

3.3.1.3 FSA

The registration state machine is illustrated in the figure below. Detailed error handling is not shown in the event actions. All events not included in the figure results in no action.

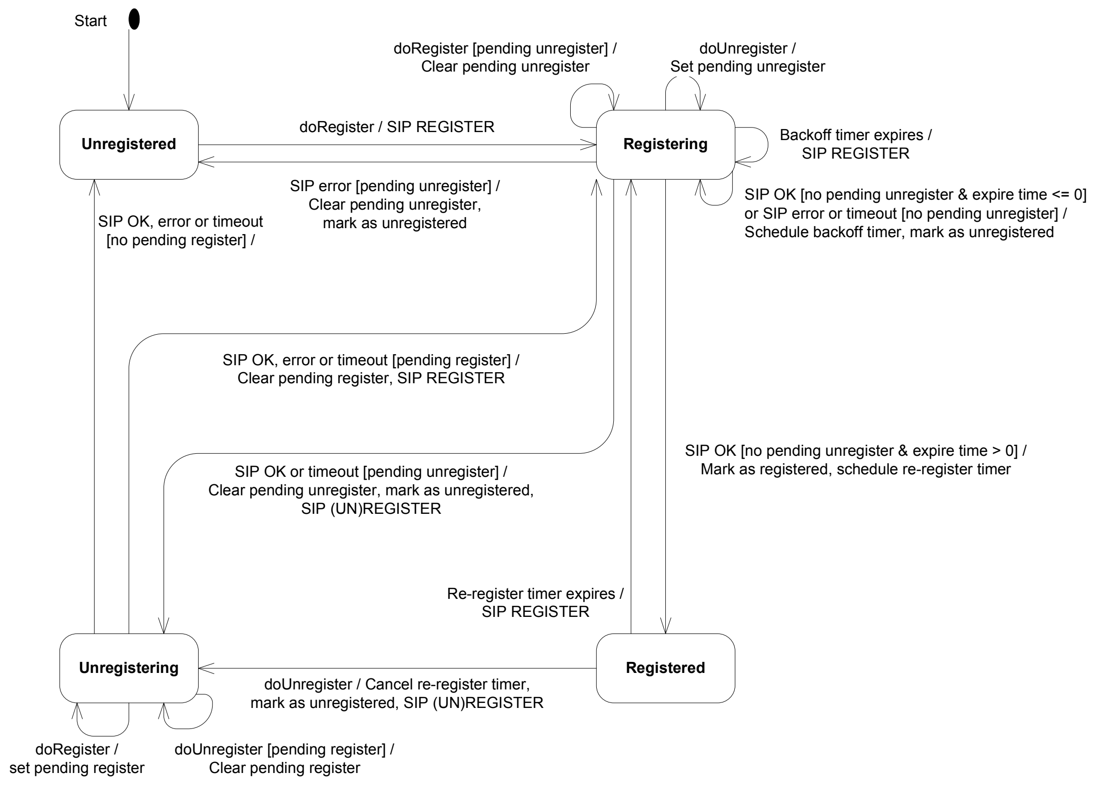

Start
Unregistered
Registering
Unregistering
Registered
doRegister [pending unregister] / Clear pending unregister
doUnregister / Set pending unregister
doRegister / SIP REGISTER
Backoff timer expires / SIP REGISTER
SIP error [pending unregister] / Clear pending unregister, mark as unregistered
SIP OK [no pending unregister & expire time <= 0] or SIP error or timeout [no pending unregister] / Schedule backoff timer, mark as unregistered
SIP OK, error or timeout [no pending register] /
SIP OK, error or timeout [pending register] / Clear pending register, SIP REGISTER
SIP OK [no pending unregister & expire time > 0] / Mark as registered, schedule re-register timer
SIP OK or timeout [pending unregister] / Clear pending unregister, mark as unregistered, SIP (UN)REGISTER
Re-register timer expires / SIP REGISTER
doUnregister / Cancel re-register timer, mark as unregistered, SIP (UN)REGISTER
doRegister / set pending register
doUnregister [pending register] / Clear pending register

Figure 6 Registration state machine

3.3.2 Registration dispatching

A registration dispatcher is used to dispatch SIP responses and timeout for the SIP REGISTER method to correct registration procedure. The dispatching is done based on an id with the following format:

```
<call id>:<from tag>
```

An example is:

```
b3688e45df9234606cd126fb009bd9d0@10.16.2.97:1837742464
```

A SIP response or timeout for the SIP REGISTER method for which an ongoing registration procedure cannot be found is ignored.

3.4 Notification

For each unsolicited SIP NOTIFY to send there is an instance of an Outbound Notification. **CM** sends out a SIP NOTIFY and awaits a response or a timeout. After the request has been answered with a final SIP response or has timed out, the Outbound Notification is removed. Any received provisional SIP response is ignored.

3.4.1 Notification dispatching

A notification dispatcher is used to dispatch SIP responses and timeout for the SIP NOTIFY method. The dispatching is done based on an id with the following format:

```
<call id>:<from tag>
```

An example is:

```
b3688e45df9234606cd126fb009bd9d0@10.16.2.97:1837742464
```

A SIP response or timeout for the SIP NOTIFY method for which an ongoing notification procedure cannot be found is ignored.

3.5 Call handling

The call handling part of **CM** is responsible for active call management. It mainly consists of a call dispatcher and zero or more calls (inbound or outbound).

CM has a state machine for each call (inbound or outbound) to keep track of the current call state.

The names of states and fired events are chosen from the CCXML specification (see [7]) so that there shall be a consistency between **CM** and a CCXML engine. **CM** has been designed so that **CM** and a CCXML engine always are in the same state. This is for example the reason why there are separate states for a call that fails or is disconnected during call setup (*Failed* state), a call that experiences an unexpected error (*Error* state) and a call that fails or is disconnected after call setup completed (*Disconnected* state). It is recommended that if new states are introduced, they should be "CCXML compliant".

NOTE: There are some rules regarding in what order actions are performed when an event is received in a call state machine:

- Due to the fact that **CM** when joining/unjoining calls looks at the state of the calls during normal execution it is important that any state transition is performed before firing an external event.
- When deleting streams, make sure that external events are fired first before the streams are deleted. This is to make sure that when a call is disconnected normally, the Disconnected event shall be received by the service before any possible event related to the deleted stream.

3.5.1 Inbound call state machine

This section describes the state machine for an inbound call.

3.5.1.1 States

The table below lists the states that exist for the inbound call state machine.

Table 7 States in inbound call state machine

State	Sub state	Description
Idle	-	The *Idle* state is the starting point to the state machine. In this state only the session creating SIP INVITE is allowed.

State	Sub state	Description
Alerting	New Call	The *Alerting-New Call* state is entered when a session creating SIP INVITE has been received and handled. In this state, the service has been loaded, a SIP 100 "Trying" response has been sent and an Alerting event has been fired.
	Early Media	The *Alerting-Early Media* state is entered when the service indicates that there is early media to play for the inbound call. In this state, a SIP 183 "Session Progress" response has been sent and an Early Media Available event has been fired.
	Early Media Wait For Prack	The *Alerting-Early Media Wait for Prack* state is entered when the service indicates that there is early media to play for the inbound call. In this state, a SIP 183 "Session Progress" response has been sent reliably and **CM** is waiting for a SIP PRACK before firing an Early Media Available event.
	Wait For Prack	The *Alerting-Wait For Prack* state is entered when the service accepts the call and provisional responses shall be sent reliably. In this state, a SIP 180 "Ringing" response has been sent reliably, and **CM** is waiting for a SIP PRACK before continuing with the call setup.
	Accepting	The *Alerting-Accepting* state is entered when the service accepts the call. In this state, a SIP 180 "Ringing" and a SIP 200 "OK" response has been sent. If the "Ringing" response was sent reliably, the SIP PRACK request has also been received and answered. In this state **CM** is waiting for a SIP ACK before the call becomes connected.
Connected	-	The *Connected* state is entered when a SIP ACK is received.
Failed	Waiting for Ack	The *Failed-Waiting for Ack* state is entered when a call in *Alerting-Accepting* state is disconnected. A SIP BYE request cannot be sent until a SIP ACK has been received.
	Lingering Bye	The *Failed-Lingering Bye* state is entered when a call in *Alerting-Accepting* state fails due to unsupported media in SIP ACK. In this state a SIP BYE request has been sent and **CM** is waiting for a response before the call is completed.
	Completed	The *Failed-Completed* state is entered for example when a SIP response is received for a SIP BYE request in *Failed-Lingering Bye* state. No SIP procedures are active and the call is completed.
Disconnected	Lingering Bye	The *Disconnected-Lingering Bye* state is entered for example when a call in *Connected* state is disconnected by the service. In this state a SIP BYE request has been sent and **CM** is waiting for a response before the call is completed.
	Completed	The *Disconnected-Completed* state is entered for example when a SIP response is received for a SIP BYE request in *Disconnected-Lingering Bye* state. No SIP procedures are active and the call is completed.
Error	Lingering Bye	The *Error-Lingering Bye* state is entered when an internal error has occurred in *Connected* state. In this state a SIP BYE request has been sent and **CM** is waiting for a response before the call is completed.

State	Sub state	Description
	Completed	The *Error-Completed* state is entered for example when a SIP response is received for a SIP BYE request in *Error-Lingering Bye* state. No SIP procedures are active and the call is completed.

3.5.1.2 Events

The table below lists the events that exist for the inbound call state machine.

Table 8 Events in inbound call state machine

Event	Description
CloseForced	O&M has requested to force close **CM**. The call shall be immediately terminated.
Play	The service requests to play media on the outbound stream.
Record	The service requests to record media from the inbound stream.
Stop play	The service requests to stop an ongoing play.
Stop record	The service requests to stop an ongoing record.
VFU request	A request to send a Video Fast Update request using SIP INFO has been received.
Accept	The service accepts the call.
Negotiate early media	The service indicates that there is early media to play for the call.
Reject	The service rejects the call.
Disconnect	The service disconnects the call.
SIP ACK	A SIP ACK request is received.
SIP BYE	A SIP BYE request is received.
SIP CANCEL	A SIP CANCEL request is received.
SIP INVITE	A session creation SIP INVITE request is received.
SIP Re-INVITE	A SIP re-INVITE request is received.
SIP OPTIONS	A SIP OPTIONS request is received.
SIP INFO	A SIP INFO request is received.
SIP PRACK	A SIP PRACK request is received.
SIP response	A SIP response is received.
SIP timeout	A SIP timeout has occurred.
"Not Accepted" timeout	The service has not accepted the call within the specified time limit.
Expires timeout	The service has not accepted the call before the Expires time (received in the SIP INVITE) was been exceeded.
"No Ack" timeout	An ACK or PRACK request was not received within the specified time limit.

Event	Description
Abandoned stream	The inbound stream has been silent for a configured amount of time. The call is considered abandoned.

3.5.1.3 FSA

The inbound call state machine is illustrated in the figure below. The figure only contains a few state transitions for a better understanding of the different states and how the interact. For complete state machine coverage, see the tables below.

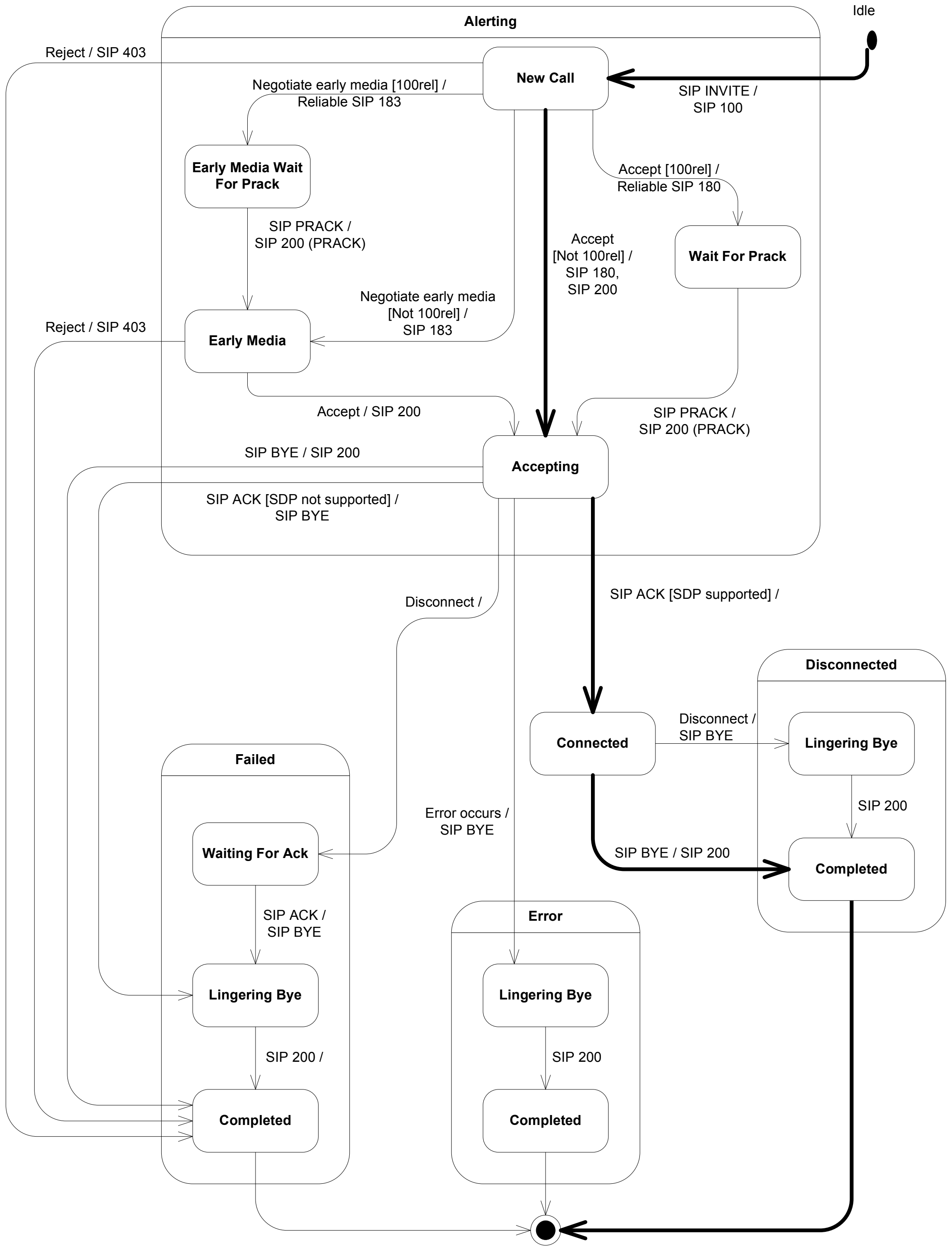


Figure 7 Inbound Call state machine

The path taken in the state machine illustrated above differs depending on if provisional SIP responses shall be sent reliably or not (indicated above with 100rel which is the SIP extension used when sending provisional responses reliably). Whether to send provisional responses reliably or not is determined as described in section 3.11.

Before a SIP request is received at this FSA, it has been validated as described in section 3.6. If the request is not valid it is rejected and will not reach this FSA.

For clarity, a received SIP response is in the tables below denoted with the following format:

`SIP <response type> (method)`

If the method is not of importance, it is not included. Examples:

SIP 2xx (BYE)	– An ok final response has been received for the SIP BYE request.
SIP 408, 481	– A SIP 408 or SIP 481 has been received for any SIP request.

If a SIP message cannot be sent by CM to the SIP stack, the call is considered lost. The state is set to *Error-Completed*, an Error event is fired and the streams are deleted. This is general error handling behavior in all states and is not shown in the tables below.

All events not included in the tables below results in no action or at most a log.

Apart from what is described above and below, setting the state to *Disconnected-Completed*, *Failed-Completed* or *Error-Completed* will cause the call to be removed from the call dispatcher and the administrative state machine.

Note that a call that is joined when its streams are deleted will be implicitly unjoined.

3.5.1.3.1 Idle state

The table below lists the event handling in the *Idle* state.

Table 9 Event handling in the Idle state

Event	Action	Next State
SIP INVITE	Send SIP 100 "Trying" response. Parse SDP offer (if any). Load service and set session data. Register to receive events. Start "Not Accepted" timer. Start Expires timer (if any). Fire Alerting event.	Alerting-New Call
Play	Fire Play Failed event	-
Record	Fire Record Failed event	-

Apart from the action described in the table above, when a SIP INVITE is received the SIP stack automatically generates a SIP 100 "Trying" response if **CM** has not within 200 milliseconds. Thus, in some situations two SIP 100 "Trying" responses might be sent for an inbound SIP INVITE.

The table below lists the error scenarios handled in the *Idle* state.

Table 10 Error handling in the Idle state

Event	Error Scenario	Action	Next State
SIP INVITE	Invalid SDP	Fire Failed event. Send SIP 488 "Not Acceptable Here" response.	Failed-Completed
SIP INVITE	Service not loaded	Fire Error event Send SIP 500 "Server Internal Error" response.	Error-Completed

3.5.1.3.2 Alerting state

The table below lists the event handling common for all sub states in the *Alerting* state.

Table 11 Event handling in Alerting state

Event	Action	Next State
Record	Fire Record Failed event.	-
SIP Re-INVITE	Send SIP 491 "Request Pending" response.	-
SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP INFO	Send SIP 405 "Method Not Allowed" response.	-

Alerting-New Call state

The table below lists the event handling in the *Alerting-New Call* state.

Table 12 Event handling in Alerting-New Call state

Event	Action	Next State
CloseForced	Fire Failed event. Send SIP 503 "Service Unavailable" response.	Failed-Completed
Play	Fire Play failed event.	-
Accept	If all responses shall be sent reliably (see section 3.11 on when to send responses reliably): If no SDP offer received in SIP INVITE: - Retrieve call type from Call-Info header or configuration. - Create inbound stream. - Create SDP offer. - Send SIP 180 "Ringing" response reliably with sdp. Otherwise: - Send SIP 180 "Ringing" response reliably, no sdp.	Alerting-Wait For Prack

Event	Action	Next State
	Otherwise: Send SIP 180 "Ringing" response. If no SDP offer received in SIP INVITE: - Retrieve call type from Call-Info header or configuration. - Create inbound stream. - Create SDP offer. Otherwise: - Retrieve SDP intersection. - Create streams. - Create SDP answer. Send SIP 200 "OK" response.	Alerting-Accepting
Negotiate early media	If no SDP offer received in SIP INVITE: Fire Early Media Failed event.	-
	Otherwise, if no responses shall be sent reliably: Retrieve SDP intersection. Create streams. Create SDP answer. Send SIP 183 "Session Progress" response. Fire Early Media Available event.	Alerting-Early Media
	Otherwise: Retrieve SDP intersection. Create streams. Create SDP answer. Send SIP 183 "Session Progress" response reliably.	Alerting-Early Media Wait For Prack
Reject	Fire Failed event. Send SIP 403 "Forbidden" response.	Failed-Completed
Expires timeout	Fire Failed event. Send SIP 487 "Request Terminated" response.	Failed-Completed
Disconnect	Fire Failed event. Fire Disconnected event. Send SIP 487 "Request Terminated" response.	Failed-Completed
SIP BYE, SIP CANCEL	Fire Failed event. Send SIP 487 "Request Terminated" response for SIP INVITE. Send SIP 200 "OK" response for SIP BYE/CANCEL.	Failed-Completed
SIP PRACK	Send SIP 403 "Forbidden" response for SIP PRACK.	-
SIP Timeout	Fire Error event.	Error-Completed
"Not Accepted" timeout	Fire Failed event. Send SIP 408 "Request Timeout" response.	Failed-Completed

The table below lists the error scenarios handled in the *Alerting-New Call* state.

Table 13 Error handling in Alerting-New Call state

Event	Error Scenario	Action	Next State
Accept, Negotiate early media	Stream or SDP not created	Fire Error event. Send SIP 500 "Server Internal Error" response.	Error-Completed
Accept, Negotiate early media	SDP intersection not found	Fire Failed event. Send SIP 488 "Not Acceptable Here" response.	Failed-Completed

Alerting-Early Media Wait For Prack state

The table below lists the event handling in the *Alerting-Early Media Wait For Prack* state.

Table 14 Event handling in Alerting-Early Media Wait For Prack state

Event	Action	Next State
CloseForced	Fire Failed event. Delete streams. Send SIP 503 "Service Unavailable" response.	Failed-Completed
Play	Fire Play failed event.	-
Accept, Negotiate early media	Fire Not Allowed event.	-
Reject	Fire Failed event. Delete streams. Send SIP 403 "Forbidden" response.	Failed-Completed
Disconnect	Fire Failed event. Fire Disconnected event. Delete streams. Send SIP 487 "Request Terminated" response.	Failed-Completed
SIP BYE, SIP CANCEL	Fire Failed event. Delete streams. Send SIP 487 "Request Terminated" response for SIP INVITE. Send SIP 200 "OK" response for SIP BYE/CANCEL.	Failed-Completed
SIP PRACK	If PRACK contains a new SDP offer, parse it and create an SDP answer. Send SIP 200 "OK" response for SIP PRACK. Fire Early Media Available event.	Alerting-Early Media
SIP Timeout	Fire Error event. Delete streams. Send SIP 504 "Server time-out" for SIP INVITE.	Error-Completed

Event	Action	Next State
Expires timeout	Fire Failed event. Delete streams. Send SIP 487 "Request Terminated" response.	Failed-Completed
"Not Accepted" timeout	Fire Failed event. Delete streams. Send SIP 408 "Request Timeout" response.	Failed-Completed
"No Ack" timeout	Fire Error event. Delete streams. Send SIP 504 "Server time-out" for SIP INVITE.	Error-Completed

The table below lists the error scenarios handled in the *Alerting-Early Media Wait For Prack* state.

Table 15 Error handling in Alerting-Early Media Wait For Prack state

Event	Error Scenario	Action	Next State
SIP PRACK	Error in SDP or SDP not acceptable.	Fire Failed event. Send SIP 488 "Not Acceptable Here" response.	Failed-Completed

Alerting-Early Media state

The table below lists the event handling in the *Alerting-Early Media* state.

Table 16 Event handling in Alerting-Early Media state

Event	Action	Next State
CloseForced	Fire Failed event. Delete streams. Send SIP 503 "Service Unavailable" response.	Failed-Completed
Play	Play media on outbound stream.	-
Stop play	Stop playing media on outbound stream.	-
Accept	Send SIP 200 "OK" response.	Alerting-Accepting
Negotiate early media	Fire Not Allowed event.	-
Reject	Fire Failed event. Delete streams. Send SIP 403 "Forbidden" response.	Failed-Completed
Disconnect	Fire Failed event. Fire Disconnected event. Delete streams. Send SIP 487 "Request Terminated" response.	Failed-Completed

Event	Action	Next State
SIP BYE, SIP CANCEL	Fire Failed event. Delete streams. Send SIP 487 "Request Terminated" response for SIP INVITE. Send SIP 200 "OK" response for SIP BYE/CANCEL.	Failed-Completed
SIP PRACK	Send SIP 403 "Forbidden" response for SIP PRACK.	-
SIP Timeout	Fire Error event. Delete streams.	Error-Completed
Expires timeout	Fire Failed event. Delete streams. Send SIP 487 "Request Terminated" response.	Failed-Completed
"Not Accepted" timeout	Fire Failed event. Delete streams. Send SIP 408 "Request Timeout" response.	Failed-Completed

The table below lists the error scenarios handled in the *Alerting-Early Media* state.

Table 17 Error handling in Alerting-Early Media state

Event	Error Scenario	Action	Next State
Play	Error playing media.	Fire Play Failed event.	-

Alerting-Wait For Prack state

The table below lists the event handling in the *Alerting-Wait For Prack* state.

Table 18 Event handling in Alerting-Wait For Prack state

Event	Action	Next State
CloseForced	Fire Failed event. Delete streams. Send SIP 503 "Service Unavailable" response.	Failed-Completed
Play	Fire Play failed event.	-
Accept, Negotiate early media	Fire Not Allowed event.	-
Reject	Fire Failed event. Delete streams. Send SIP 403 "Forbidden" response.	Failed-Completed
Disconnect	Fire Failed event. Fire Disconnected event. Delete streams. Send SIP 487 "Request Terminated" response.	Failed-Completed

Event	Action	Next State
SIP BYE, SIP CANCEL	Fire Failed event. Delete streams. Send SIP 487 "Request Terminated" response for SIP INVITE. Send SIP 200 "OK" response for SIP BYE/CANCEL.	Failed-Completed
SIP PRACK	Send SIP 200 "OK" response for SIP PRACK. If no SDP offer received in SIP INVITE: - Retrieve call type from Call-Info header or configuration. - Create inbound stream. - Create SDP offer. Otherwise: - Retrieve SDP intersection. - Create streams. - Create SDP answer. Send SIP 200 "OK" response for SIP INVITE.	Alerting-Accepting
SIP Timeout	Fire Error event. Delete streams. Send SIP 504 "Server time-out" for SIP INVITE.	Error-Completed
Expires timeout	Fire Failed event. Delete streams. Send SIP 487 "Request Terminated" response.	Failed-Completed
"No Ack" timeout	Fire Error event. Delete streams. Send SIP 504 "Server time-out" for SIP INVITE.	Error-Completed

The table below lists the error scenarios handled in the *Alerting-Wait For Prack* state.

Table 19 Error handling in Alerting-Wait For Prack state

Event	Error Scenario	Action	Next State
SIP PRACK	Stream or SDP not created	Fire Error event. Send SIP 500 "Server Internal Error" response for SIP INVITE.	Error-Completed
SIP PRACK	SDP intersection not found	Fire Failed event. Send SIP 488 "Not Acceptable Here" response for SIP INVITE.	Failed-Completed

Alerting-Accepting state

The table below lists the event handling in the *Alerting-Accepting* state.

Table 20 Event handling in Alerting-Accepting state

Event	Action	Next State

Event	Action	Next State
CloseForced	Fire Failed event. Delete streams.	Failed-Waiting For Ack
Disconnect	Fire Failed event. Fire Disconnected event. Delete streams.	Failed-Waiting For Ack
Play	Play media on outbound stream.	-
Stop play	Stop playing media on outbound stream.	-
Accept, Reject, Negotiate early media	Fire Not Allowed event.	-
SIP ACK	If no SDP offer received in SIP INVITE: Parse SDP. Retrieve SDP intersection. Create outbound stream. Fire Connected event.	Connected
SIP BYE	Fire Failed event. Delete streams. Send SIP 200 "OK" response.	Failed-Completed
SIP CANCEL	Send SIP 200 "OK" response.	-
SIP PRACK	Send SIP 403 "Forbidden" response for SIP PRACK.	-
SIP Timeout	Fire Error event. Delete streams. Send SIP BYE request.	Error-Lingering Bye
"No Ack" timeout	Fire Error event. Delete streams.	Error-Completed

The table below lists the error scenarios handled in the *Alerting-Accepting* state.

Table 21 Error handling in Alerting-Accepting state

Event	Condition	Action	Next State
SIP ACK	Stream not created, invalid SDP	Fire Error event. Delete streams. Send SIP BYE request.	Error-Lingering Bye
SIP ACK	No SDP answer when expected, no SDP intersection found, incompatible call type in SDP	Fire Failed event. Delete streams. Send SIP BYE request.	Failed-Lingering Bye

3.5.1.3.3 Connected state

The table below lists the event handling in the *Connected* state.

Table 22 Event handling in Connected state

Event	Action	Next State
CloseForced, Disconnect, SIP 408 or 481 (INFO), Abandoned stream	Fire Disconnected event. Delete streams. Send SIP BYE request.	Disconnected-Lingering Bye
Play	Play media on outbound stream.	-
Record	Record media from inbound stream.	-
Stop Play	Stop playing media on outbound stream.	-
Stop Record	Stop recording media from inbound stream.	-
VFU request	Send SIP INFO request.	-
Accept, Reject, Negotiate early media	Fire Not Allowed event.	-
SIP BYE	Fire Disconnected event. Delete streams. Send SIP 200 "OK" response.	Disconnected-Completed
SIP CANCEL, SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP INFO	If the call is joined, forward the SIP request to the other call. Otherwise, send SIP 405 "Method Not Allowed" response.	-
SIP Re-INVITE	Send SIP 488 "Not Acceptable Here" response.	-
SIP PRACK	Send SIP 403 "Forbidden" response for SIP PRACK.	-
SIP not 408 or 481 (INFO)	If the call is joined, forward the SIP response to the other call. Otherwise, parse media control information.	-
SIP timeout	If configured to disconnect on SIP timeout: Fire Error event, Delete streams.	Error-Completed

The table below lists the error scenarios handled in the *Connected* state.

Table 23 Error handling in Connected state

Event	Error Scenario	Action	Next State
Play	Error playing media.	Fire Play Failed event.	-
Record	Error recording media.	Fire Record Failed event.	-

3.5.1.3.4 Failed state

The table below lists the event handling common for all sub states in the *Failed* state.

Table 24 Event handling in Failed state

Event	Action	Next State
Play	Fire Play Failed event.	-
Record	Fire Record Failed event.	-
Accept, Reject, Negotiate early media	Fire Not Allowed event.	-
Disconnect	Fire Disconnected event.	-
SIP CANCEL, SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP PRACK	Send SIP 403 "Forbidden" response for SIP PRACK.	-
SIP Timeout	Fire Error event. Delete streams.	Error-Completed
SIP response (INFO)	Parse media control information.	-

Failed-Completed state

The table below lists the event handling in the *Failed-Completed* state.

Table 25 Event handling in Failed-Completed state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	-
SIP Re-INVITE	Send SIP 488 "Not Acceptable Here" response.	-

Failed-Lingering Bye state

The table below lists the event handling in the *Failed-Lingering* Bye state.

Table 26 Event handling in Failed-Lingering Bye state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	Failed-Completed
SIP response (BYE)		Failed-Completed

Failed-Waiting for Ack state

The table below lists the event handling in the *Failed-Waiting for Ack* state.

Table 27 Event handling in Failed-Waiting for Ack state

Event	Action	Next State

Event	Action	Next State
SIP ACK	Send SIP BYE request.	Failed-Lingering Bye
SIP BYE	Send SIP 200 "OK" response.	Failed-Completed
SIP Re-INVITE	Send SIP 491 "Request Pending" response.	-
"No Ack" timeout		Failed-Completed

3.5.1.3.5 Disconnected state

The table below lists the event handling common for all sub states in the *Disconnected* state.

Table 28 Event handling in Disconnected state

Event	Action	Next State
Play	Fire Play Failed event.	-
Record	Fire Record Failed event.	-
Accept, Reject, Negotiate early media	Fire Not Allowed event.	-
Disconnect	Fire Disconnected event.	-
SIP CANCEL, SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP Re-INVITE	Send SIP 488 "Not Acceptable Here" response.	-
SIP PRACK	Send SIP 403 "Forbidden" response for SIP PRACK.	-
SIP Timeout	Fire Error event. Delete streams.	Error-Completed
SIP response (INFO)	Parse media control information.	-

Disconnected-Completed state

The table below lists the event handling in the *Disconnected-Completed* state.

Table 29 Event handling in Disconnected-Completed state

Event	Action	Next State
BYE	Send SIP 200 "OK" response.	-

Disconnected-Lingering Bye state

The table below lists the event handling in the *Disconnected-Lingering Bye* state.

Table 30 Event handling in Disconnected-Lingering Bye state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	Disconnected-Completed
SIP response (BYE)		Disconnected-Completed

3.5.1.3.6 Error state

The table below lists the event handling common for all sub states in the *Error* state.

Table 31 Event handling in Error state

Event	Action	Next State
Play	Fire Play Failed event.	-
Record	Fire Record Failed event.	-
Accept, Reject, Negotiate early media	Fire Not Allowed event.	-
Disconnect	Fire Disconnected event.	-
SIP CANCEL, SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP Re-INVITE	Send SIP 488 "Not Acceptable Here" response.	-
SIP PRACK	Send SIP 403 "Forbidden" response for SIP PRACK.	-
SIP Timeout	Fire Error event. Delete streams.	Error-Completed
SIP response (INFO)	Parse media control information.	-

Error-Completed state

The table below lists the event handling in the *Error-Completed* state.

Table 32 Event handling in Error-Completed state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	-

Error-Lingering Bye state

The table below lists the event handling in the *Error-Lingering* Bye state.

Table 33 Event handling in Error-Lingering Bye state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	Error-Completed

Event	Action	Next State
SIP response (BYE)		Error-Completed

3.5.2 Outbound call state machine

This section describes the state machine for an outbound call.

3.5.2.1 States

The table below lists the states that exist for the outbound call state machine.

Table 34 States in outbound call state machine

State	Sub state	Description
Idle	-	The *Idle* state is the starting point to the state machine. In this state only the service command dial is allowed.
Progressing	Calling	The *Progressing-Calling* state is entered when a session creating SIP INVITE has been sent.
	Proceeding	The *Progressing-Proceeding* state is entered when a provisional SIP response (not containing early media indication) is received for the SIP INVITE. In this state a Progressing event may have been fired (depending on the type of response).
	Early Media	The *Progressing-Early Media* state is entered when a SIP 183 "Session Progress" response containing an SDP has been received for the INVITE. In this state a Progressing event indicating early media has been fired.
Connected	-	The *Connected* state is entered when a SIP 200 "OK" response is received. In this state a SIP ACK response has been sent.
Failed	Waiting for Response	The *Failed-Waiting for Response* state is entered when a call in *Progressing-Calling* state is disconnected. A SIP CANCEL or BYE request cannot be sent until a SIP response has been received.
	Lingering Bye	The *Failed-Lingering Bye* state is entered for example when a call in *Progressing-Proceeding* state fails due to unsupported media in SIP 200 "OK". In this state a SIP BYE request has been sent and **CM** is waiting for a response before the call is completed.
	Lingering Cancel	The *Failed-Lingering Cancel* state is entered for example when a call in *Progressing-Proceeding* state fails due to unsupported media in SIP 183 "Session Progress". In this state a SIP CANCEL request has been sent and **CM** is waiting for a response before the call is completed.
	Completed	The *Failed-Completed* state is entered for example when a SIP response is received for a SIP BYE request in *Failed-Lingering Bye* state. No SIP procedures are active and the call is completed.
Disconnected	Lingering Bye	The *Disconnected-Lingering Bye* state is entered for example when a call in *Connected* state is disconnected by the service. In this state a SIP BYE request has been sent and **CM** is waiting for a response before the call is completed.

State	Sub state	Description
	Completed	The *Disconnected-Completed* state is entered for example when a SIP response is received for a SIP BYE request in *Disconnected-Lingering Bye* state. No SIP procedures are active and the call is completed.
Error	Lingering Bye	The *Error-Lingering Bye* state is entered when an internal error has occurred in *Connected* state. In this state a SIP BYE request has been sent and **CM** is waiting for a response before the call is completed.
	Lingering Cancel	The *Error-Lingering Cancel* state is entered for example when an internal error has occurred in *Progressing-Proceeding* state. In this state a SIP CANCEL request has been sent and **CM** is waiting for a response before the call is completed.
	Completed	The *Error-Completed* state is entered for example when a SIP response is received for a SIP BYE request in *Error-Lingering Bye* state. No SIP procedures are active and the call is completed.

3.5.2.2 Events

The table below lists the events that exist for the outbound call state machine.

Table 35 Events in outbound call state machine

Event	Description
CloseForced	O&M has requested to force close **CM**. The call shall be immediately terminated.
Play	The service requests to play media on the outbound stream.
Record	The service requests to record media from the inbound stream.
Stop play	The service requests to stop an ongoing play.
Stop record	The service requests to stop an ongoing record.
VFU request	A request to send a Video Fast Update request using SIP INFO has been received.
Disconnect	The service disconnects the call.
Dial	The service initiates the outbound call.
Send token	The service whishes to send a DTMF token on the outbound media stream.
SIP ACK	A SIP ACK request is received.
SIP BYE	A SIP BYE request is received.
SIP CANCEL	A SIP CANCEL request is received.
SIP Re-INVITE	A SIP re-INVITE request is received.
SIP OPTIONS	A SIP OPTIONS request is received.
SIP INFO	A SIP INFO request is received.
SIP PRACK	A SIP PRACK request is received.
SIP response	A SIP response is received.

Event	Description
SIP timeout	A SIP timeout has occurred.
"Max Duration" timeout	The maximum duration for a connected call has been reached. The call is disconnected.
"Not Connected" timeout	The outbound call has not been connected within the specified time limit.
"No Ack" timeout	An ACK request was not received for a SIP re-INVITE within the specified time limit.
"No response" timeout	A SIP response was not received within the specified time limit.
Abandoned stream	The inbound media stream has been silent for a configured amount of time. The call is considered abandoned.

3.5.2.3 FSA

The outbound call state machine is illustrated in the figure below. The figure only contains a few state transitions for a better understanding of the different states and how the interact. For complete state machine coverage, see the tables below.

Note that SIP ACK requests for error responses are generated automatically by the SIP stack and are therefore not included in the figure or in the tables below.

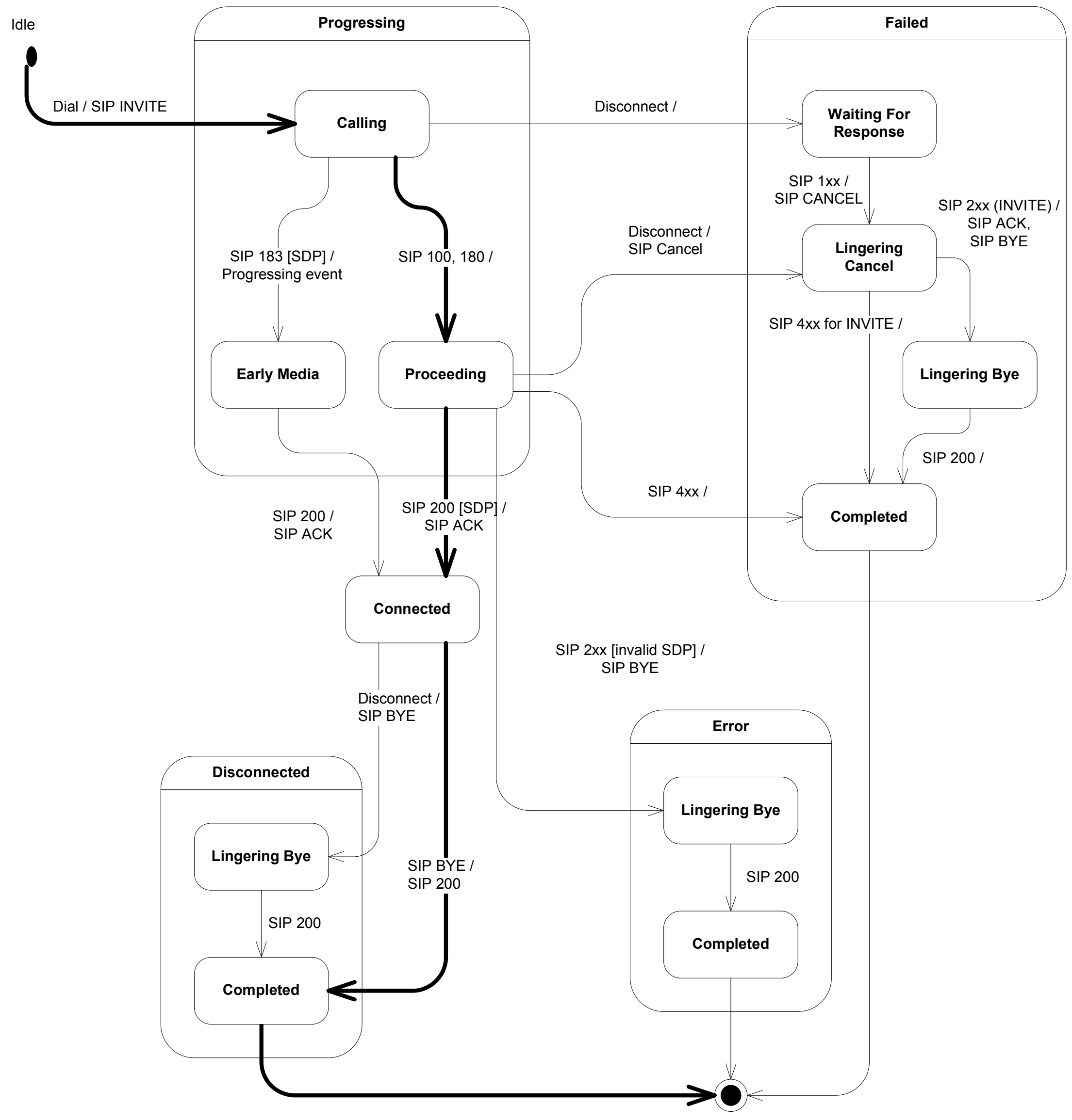


Figure 8 Outbound call state machine

Before a SIP request is received at this FSA, it has been validated as described in section 3.6. If the request is not valid it is rejected and will not reach this FSA.

For clarity, a received SIP response is in the tables below denoted with the following format:

SIP <response type> (method)

If the method is not of importance, it is not included. Examples:

SIP 2xx (BYE) – An ok final response has been received for the SIP BYE request.
SIP 408, 481 – A SIP 408 or SIP 481 has been received for any SIP request.

If a SIP message cannot be sent by CM to the SIP stack, the call is considered lost. The state is set to *Error-Completed*, an Error event is fired and the streams are deleted. This is general error handling behavior in all states and is not shown in the tables below.

All events not included in the tables below results in no action or at most a log.

Apart from what is described above and below, setting the state to *Disconnected-Completed*, *Failed-Completed* or *Error-Completed* will cause the call to be removed from the call dispatcher and the administrative state machine.

Note that a call that is joined when its streams are deleted will be implicitly unjoined.

3.5.2.3.1 Idle state

The table below lists the event handling in the *Idle* state.

Table 36 Event handling in Idle state

Event	Action	Next State
CloseForced	Fire Failed event.	Failed-Completed
Disconnect	Fire Failed event. Fire Disconnected event.	Failed-Completed
Dial	Create inbound stream. Create SDP offer. Retrieve remote party address to send SIP INVITE to. Send SIP INVITE request. Register to receive events. Start "Not Connected" timer.	Progressing-Calling
Play	Fire Play Failed event	-
Record	Fire Record Failed event	-
Send token	Fire "Not Allowed" event.	-

The table below lists the error scenarios handled in the *Idle* state.

Table 37 Error handling in Idle state

Event	Error Scenario	Action	Next State
Dial	Stream or SDP could not be created, remote party could not be retrieved or found, INVITE could not be created.	Fire Error event. Delete streams.	Error-Completed
Dial	INVITE could not be sent.	Fire Failed event Delete streams.	Failed-Completed

3.5.2.3.2 Progressing state

The table below lists the event handling common for all sub states in the *Progressing* state.

Table 38 Event handling in Progressing state

Event	Action	Next State
Play	Fire Play Failed event.	-
Dial, Send token	Fire "Not Allowed" event.	-
SIP BYE	Fire Failed event. Delete streams. SIP 200 "OK" for SIP BYE request.	Failed-Completed
SIP Re-INVITE	Send SIP 491 "Request Pending" response.	-
SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP PRACK	Send SIP 403 "Forbidden" response.	-
SIP Timeout (INVITE)	Retrieve new remote party address to send SIP INVITE to. Send new SIP INVITE request.	Progressing-Calling
SIP Timeout (PRACK)	Fire Error event. Delete streams. Send SIP CANCEL request.	Error-Lingering Cancel
SIP Timeout (Other than INVITE or PRACK)	Fire Error event. Delete streams.	Error-Completed

The table below lists the error scenarios common for all sub states in the *Progressing* state.

Table 39 Error handling in the Progressing state

Event	Error Scenario	Action	Next State
SIP Timeout (INVITE)	No new remote party could be found, INVITE could not be sent.	Fire Failed event. Delete streams.	Failed-Completed
SIP Timeout (INVITE)	INVITE could not be created.	Fire Error event Delete streams.	Error-Completed

Progressing-Calling state

The table below lists the event handling in the *Progressing-Calling* state.

Table 40 Event handling in the Progressing-Calling state

Event	Action	Next State
CloseForced, "Not Connected" timeout	Fire Failed event. Delete streams.	Failed-Waiting for Response

Event	Action	Next State
Disconnect	Fire Failed event. Fire Disconnected event. Delete streams.	Failed-Waiting for Response
Record	Fire Record Failed event.	-
SIP INFO	Send SIP 405 "Method Not Allowed" response.	-
SIP 100, 181, or 182 (INVITE)		Progressing-Proceeding
SIP 180 (INVITE), SIP 183 or unknown 1xx (INVITE) without SDP	Fire Progressing event. If response was sent reliably, send SIP PRACK request.	Progressing-Proceeding
SIP 183 or unknown 1xx (INVITE) with SDP	Retrieve SDP intersection. Create outbound stream. Fire Progressing event. If response was sent reliably, send SIP PRACK request.	Progressing- Early Media
SIP 2xx (INVITE)	Send SIP ACK request. Retrieve SDP intersection from SDP answer. Create outbound stream. Fire Connected event.	Connected
SIP 3xx (INVITE)	Retrieve contacts from response and select next contact to try. Send redirected INVITE request to new contact.	Progressing-Calling
SIP 5xx (INVITE) for redirected call	Select next contact to try from previously received contact list. Send redirected INVITE request to new contact.	Progressing-Calling
SIP 5xx (INVITE) for non redirected call, SIP 4xx, 6xx (INVITE)	Fire Failed event. Delete streams.	Failed-Completed
SIP 408, 481 (PRACK)	Fire Failed event. Delete streams. Send SIP CANCEL request.	Failed-Lingering Cancel

The table below lists the error scenarios handled in the *Progressing-Calling* state.

Table 41 Error handling in the Progressing-Calling state

Event	Error Scenario	Action	Next State
SIP 183, unknown 1xx (INVITE)	Error retrieving SDP answer, Stream could not be created	Fire Error event. Delete streams. Send SIP CANCEL request.	Error- Lingering Cancel

Event	Error Scenario	Action	Next State
SIP 183, unknown 1xx (INVITE)	No SDP intersection found	Fire Failed event. Delete streams. Send SIP CANCEL request.	Failed- Lingering Cancel
SIP 2xx (INVITE)	No SDP answer in INVITE, No SDP intersection found	Send SIP ACK request. Fire Failed event. Delete streams. Send SIP BYE request.	Failed- Lingering Bye
SIP 2xx (INVITE)	Error retrieving SDP answer, Stream could not be created	Send SIP ACK request. Fire Error event. Delete streams. Send SIP BYE request.	Error- Lingering Bye
SIP 3xx (INVITE)	Call already redirected (only one level is supported), No next contact found, INVITE could not be sent	Fire Failed event. Delete streams.	Failed- Completed.
SIP 3xx (INVITE), SIP 5xx (INVITE)	INVITE could not be created	Fire Error event. Delete streams.	Error-Completed
SIP 5xx (INVITE)	No next contact found, INVITE could not be sent	Fire Failed event. Delete streams.	Failed-Completed

Progressing-Early Media state

The table below lists the event handling in the *Progressing-Early Media* state.

Table 42 Event handling in the Progressing-Early Media state

Event	Action	Next State
CloseForced, "Not Connected" timeout, SIP 408, 481 (INFO)	Fire Failed event. Delete streams. Send SIP CANCEL request.	Failed- Lingering Cancel
Disconnect	Fire Failed event. Fire Disconnected event. Delete streams. Send SIP CANCEL request.	Failed- Lingering Cancel
Record	Record media from inbound stream.	-
Stop record	Stop recording media from inbound stream.	-
VFU request	Send SIP INFO request.	-
SIP INFO	If the call is joined, forward the SIP request to the other call. Otherwise, send SIP 405 "Method Not Allowed" response.	-

Event	Action	Next State
SIP 180, 183, unknown 1xx (INVITE)	Fire Progressing event. If response was sent reliably, send SIP PRACK request.	-
SIP 2xx (INVITE)	Send SIP ACK request. Fire Connected event.	Connected
SIP 3xx, 5xx (INVITE)	Fire Failed event. Delete streams.	Failed-Completed
SIP 408, 481 (PRACK)	Fire Failed event. Delete streams. Send SIP CANCEL request.	Failed-Lingering Cancel
SIP not 408 or 481 (INFO)	If the call is joined, forward the SIP response to the other call. Otherwise, parse media control information.	-

The table below lists the error scenarios handled in the *Progressing-Early Media* state.

Table 43 Error handling in the Progressing-Early Media state

Event	Error Scenario	Action	Next State
Record	Error when recording media.	Fire Record Failed event.	-

Progressing-Proceeding state

The table below lists the event handling in the *Progressing-Proceeding* state.

Table 44 Event handling in the Progressing-Proceeding state

Event	Action	Next State
CloseForced, "Not Connected" timeout	Fire Failed event. Delete streams. Send SIP CANCEL request.	Failed- Lingering Cancel
Disconnect	Fire Failed event. Fire Disconnected event. Delete streams. Send SIP CANCEL request.	Failed- Lingering Cancel
Record	Fire Record Failed event.	-
SIP INFO	Send SIP 405 "Method Not Allowed" response.	-
SIP 180, 183, unknown 1xx (INVITE)	Fire Progressing event. If response was sent reliably, send SIP PRACK request.	Progressing-Proceeding

Event	Action	Next State
SIP 183, unknown 1xx (INVITE) with SDP	Retrieve SDP intersection. Create outbound stream. Fire Progressing event. If response was sent reliably, send SIP PRACK request.	Progressing-Early Media
SIP 2xx (INVITE)	Send SIP ACK request. Retrieve SDP intersection from SDP. Create outbound stream. Fire Connected event.	Connected
SIP 3xx (INVITE)	Retrieve contacts from response and select next contact to try. Send redirected INVITE request to new contact.	Progressing-Calling
SIP 5xx (INVITE) for redirected call	Select next contact to try from previously received contact list. Send redirected INVITE request to new contact.	Progressing-Calling
SIP 5xx (INVITE) for a non redirected call, SIP 4xx, 6xx (INVITE)	Fire Failed event. Delete streams.	Failed-Completed
SIP 408, 481 (PRACK)	Fire Failed event. Delete streams. Send SIP CANCEL request.	Failed-Lingering Cancel

The table below lists the error scenarios handled in the *Progressing-Proceeding* state.

Table 45 Error handling in the Progressing-Proceeding state

Event	Error Scenario	Action	Next State
SIP 183, unknown 1xx (INVITE)	Error retrieving SDP answer, Stream could not be created	Fire Error event. Delete streams. Send SIP CANCEL request.	Error-Lingering Cancel
SIP 183, unknown 1xx (INVITE)	No SDP intersection found	Fire Failed event. Delete streams. Send SIP CANCEL request.	Failed-Lingering Cancel
SIP 2xx (INVITE)	No SDP answer in INVITE, No SDP intersection found	Send SIP ACK request. Fire Failed event. Delete streams. Send SIP BYE request.	Failed-Lingering Bye

Event	Error Scenario	Action	Next State
SIP 2xx (INVITE)	Error retrieving SDP answer, Stream could not be created	Send SIP ACK request. Fire Error event. Delete streams. Send SIP BYE request.	Error-Lingering Bye
SIP 3xx (INVITE)	Call already redirected (only one level is supported), No next contact found, INVITE could not be sent	Fire Failed event. Delete streams.	Failed-Completed.
SIP 3xx (INVITE), SIP 5xx (INVITE)	INVITE could not be created	Fire Error event. Delete streams.	Error-Completed
SIP 5xx (INVITE)	No next contact found, INVITE could not be sent	Fire Failed event. Delete streams.	Failed-Completed

3.5.2.3.3 Connected state

The table below lists the event handling in the *Connected* state.

Table 46 Event handling in the Connected state

Event	Action	Next State
CloseForced, Disconnect, SIP 408 or 481 (INFO), "Max Duration" timeout, Abandoned Stream	Fire Disconnected event. Delete streams. Send SIP BYE request.	Disconnected-Lingering Bye
Play	Play media on outbound stream.	-
Record	Record media from inbound stream.	-
Stop Play	Stop playing media on outbound stream.	-
Stop Record	Stop recording media from inbound stream.	-
VFU request	Send SIP INFO request.	-
Dial, Send token	Fire Not Allowed event.	-
SIP BYE	Fire Disconnected event. Delete streams. Send SIP 200 "OK" response.	Disconnected-Completed
SIP Re-INVITE	Send SIP 488 "Not Acceptable Here" response.	-
SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP INFO	If the call is joined, forward the SIP request to the other call. Otherwise, send SIP 405 "Method Not Allowed" response.	-

Event	Action	Next State
SIP PRACK	Send SIP 403 "Forbidden" response.	-
SIP response other than 408 or 481 (INFO)	If the call is joined, forward the SIP response to the other call. Otherwise, parse the received media control information in order to log any errors.	-
SIP timeout	Fire Error event, Delete streams.	Error-Completed
"No Ack" timeout	Fire Error event. Delete streams. Send SIP BYE request.	Error-Lingering Bye

The table below lists the error scenarios handled in the *Connected* state.

Table 47 Error handling in Connected state

Event	Error Scenario	Action	Next State
Play	Error playing media.	Fire Play Failed event.	-
Record	Error recording media.	Fire Record Failed event.	-

3.5.2.3.4 Failed state

The table below lists the event handling common for all sub states in the *Failed* state.

Table 48 Event handling in Failed state

Event	Action	Next State
Play	Fire Play Failed event.	-
Record	Fire Record Failed event.	-
Disconnect	Fire Disconnected event.	-
Dial, Send token	Fire Not Allowed event.	-
SIP Re-INVITE	Send SIP 488 "Not Acceptable Here" response.	-
SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP PRACK	Send SIP 403 "Forbidden" response.	-
SIP response (INFO)	Parse media control information.	-
SIP Timeout	Fire Error event. Delete streams.	Error-Completed

Failed-Completed state

The table below lists the event handling in the *Failed-Completed* state.

Table 49 Event handling in Failed-Completed state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	-

Failed-Lingering Cancel state

The table below lists the event handling in the *Failed-Lingering Cancel* state.

Table 50 Event handling in Failed-Lingering Cancel state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	Failed-Completed
SIP 2xx (INVITE)	Send SIP ACK request. Send SIP BYE request.	Failed- Lingering Bye
SIP 3xx-6xx (INVITE)		Failed-Completed

Failed-Lingering Bye state

The table below lists the event handling in the *Failed-Lingering Bye* state.

Table 51 Event handling in Failed-Lingering Bye state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	Failed-Completed
SIP response (BYE)		Failed-Completed

Failed-Waiting for Response state

The table below lists the event handling in the *Failed-Waiting for Response* state.

Table 52 Event handling in Failed-Waiting for Response state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	Failed-Completed
SIP 1xx (INVITE)	Send SIP ACK request. Send SIP CANCEL request.	Failed- Lingering Cancel
SIP 2xx (INVITE)	Send SIP ACK request. Send SIP BYE request.	Failed- Lingering Bye
SIP 3xx , 4xx (INVITE)		Failed-Completed
"No Response" timeout		Failed-Completed

3.5.2.3.5 Disconnected state

The table below lists the event handling common for all sub states in the *Disconnected* state.

Table 53 Event handling in Disconnected state

Event	Action	Next State
Play	Fire Play Failed event.	-
Record	Fire Record Failed event.	-
Disconnect	Fire Disconnected event.	-
Dial, Send token	Fire Not Allowed event.	-
SIP Re-INVITE	Send SIP 488 "Not Acceptable Here" response.	-
SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP PRACK	Send SIP 403 "Forbidden" response.	-
SIP response (INFO)	Parse media control information.	-
SIP Timeout	Fire Error event. Delete streams.	Error-Completed

Disconnected-Completed state

The table below lists the event handling in the *Disconnected-Completed* state.

Table 54 Event handling in Disconnected-Completed state

Event	Action	Next State
BYE	Send SIP 200 "OK" response.	-

Disconnected-Lingering Bye state

The table below lists the event handling in the *Disconnected-Lingering Bye* state.

Table 55 Event handling in Disconnected-Lingering Bye state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	Disconnected-Completed
SIP response (BYE)		Disconnected-Completed
SIP timeout		Error-Completed

3.5.2.3.6 Error state

The table below lists the event handling common for all sub states in the *Error* state.

Table 56 Event handling in Error state

Event	Action	Next State

Event	Action	Next State
Play	Fire Play Failed event.	-
Record	Fire Record Failed event.	-
Disconnect	Fire Disconnected event.	-
Dial, Send token	Fire Not Allowed event.	-
SIP Re-INVITE	Send SIP 488 "Not Acceptable Here" response.	-
SIP OPTIONS	Send SIP 200 "OK" response.	-
SIP PRACK	Send SIP 403 "Forbidden" response.	-
SIP response (INFO)	Parse media control information.	-
SIP Timeout	Fire Error event. Delete streams.	Error-Completed

Error-Completed state

The table below lists the event handling in the *Error-Completed* state.

Table 57 Event handling in Error-Completed state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	-

Error-Lingering Bye state

The table below lists the event handling in the *Error-Lingering Bye* state.

Table 58 Event handling in Error-Lingering Bye state

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	Error-Completed
SIP response (BYE)		Error-Completed

Error-Lingering Cancel state

The table below lists the event handling in the *Error-Lingering Cancel* state.

Event	Action	Next State
SIP BYE	Send SIP 200 "OK" response.	Error-Completed
SIP 2xx (INVITE)	Send SIP ACK request. Send SIP BYE request.	Error- Lingering Bye
SIP 3xx-6xx (INVITE)		Error-Completed

3.5.3 Call dispatching

The call dispatching is done based on the dialog ID. The dialog ID consists of a Call-ID value, a local tag and a remote tag.

The dialog ID varies over time when a new dialog is setup. For example, for inbound calls the local tag is not set until the first non-trying (i.e. not SIP 100) response is generated. This means that between the initial INVITE and the first non-trying response the dialog ID does not include the local tag. Also, when the local tag has been set in the response there is still a possibility that SIP messages (e.g. a SIP CANCEL) arrives for the call without the tag set due to timing issues.

Thus, the call dispatcher must be able to handle different dialog IDs mapping to the same call for a period of time.

When an initial INVITE is received, the early dialog ID (without local tag) is mapped to the new call. Later, when the first non-trying response is sent the updated dialog ID (with local tag) is also mapped to the call. Finally, when the first SIP message is received that matches the updated dialog ID (with local tag), the mapping for the early dialog ID to the call is removed.

As a safety net, if a matching call is not found based on the dialog ID, or no dialog ID is found, the dispatcher creates an own ID and tries to match the call based on this. The format of the created ID differs for client and server transactions.

The ID format for a client transaction:

```
<call id>:<from tag>
```

The ID format for a server transaction:

```
<call id>:<to tag>
```

A SIP response or timeout sent within dialog for which an active call cannot be found is ignored.

A SIP request send within dialog for which an active call cannot be found is responded with a SIP 481 "Call/Transaction Does Not Exist".

3.6 SIP message reception

A SIP response or SIP timeout received by **CM** is dispatched to a corresponding register procedure if the SIP method is REGISTER or otherwise to a corresponding call. If no register procedure or call is found, the SIP response or timeout is ignored.

A SIP request received by **CM** is handled as illustrated in the figure below.

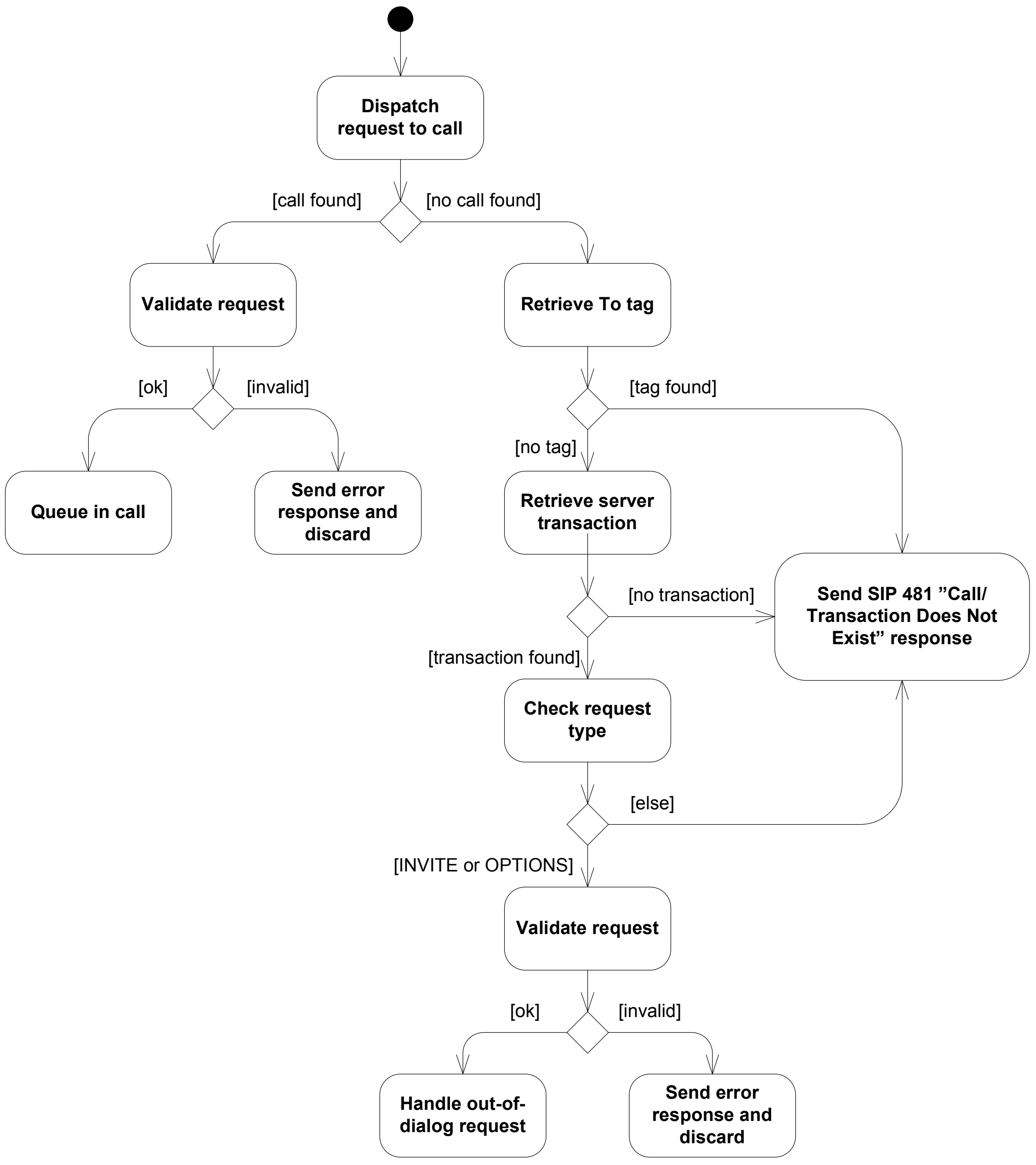


Figure 9 Handling of SIP requests

The request validation is done as described in the table below.

Table 59 Sip request validation

Condition	Action
Request method is known but not allowed, i.e. REGISTER or INFO	Send SIP 405 "Method Not Allowed" response.
Request method is unknown, i.e. differs from INVITE, ACK, BYE, CANCEL, OPTIONS, REGISTER or INFO	Send SIP 501 "Not Implemented" response.

Condition	Action
To-header is invalid.	Send SIP 403 "Forbidden" response.
Unsupported URI scheme in Request-URI.	Send SIP 416 "Unsupported URI Scheme" response.
Request-URI is invalid.	Send SIP 404 "Not Found" response.
Unsupported extension is required.	Send SIP 420 "Bad Extension" response.
Unsupported Content-Language, Content-Encoding or Content-Type.	Send SIP 415 "Unsupported Media Type" response.

For information on what is not supported, see [2].

The figure and table above lists situations in which a SIP error response is sent for the request. If the request is a SIP ACK a response is not sent. The SIP ACK is simply ignored.

Only SIP INVITE and SIP OPTIONS are handled out-of-dialog. An out-of-dialog SIP INVITE results in a new inbound call being created (dependant of course on the administrative and load regulation states). An out-of-dialog SIP OPTIONS is handled in the same way as the INVITE with the exception that no new inbound call is created.

3.7 SIP stack configuration

The table below lists how the SIP stack is configured by **CM**.

Table 60 SIP stack configuration

Stack property	Value
javax.sip.IP_ADDRESS	Host name given when **CM** service is initialized.
javax.sip.STACK_NAME	"NISTv1.2"
javax.sip.RETRANSMISSION_FILTER	true (Indicates that the SIP stack should handle retransmissions normally handled in the forth SIP layer; Transaction User.)
gov.nist.javax.sip.TRACE_LEVEL	0
gov.nist.javax.sip.CACHE_SERVER_CONNECTIONS	false (Caching boosts performance but opens up to possible TCP based Denial of Service attacks. It is therefore turned off.)
gov.nist.javax.sip.CACHE_CLIENT_CONNECTIONS	false (Caching boosts performance but opens up to possible TCP based Denial of Service attacks. It is therefore turned off.)
gov.nist.javax.sip.READ_TIMEOUT	1000 (This is a guard against TCP read starvation.)

Stack property	Value
gov.nist.javax.sip.REENTRANT_LISTENER	true (Indicates that the listener (in this case **CM**) is re-entrant. It improves performance.)
gov.nist.javax.sip.THREAD_POOL_SIZE	8 (Limits the thread pool size to improve performance.)
gov.nist.javax.sip.T2	Configured value.
gov.nist.javax.sip.T4	Configured value.
gov.nist.javax.sip.TIMER_B	Configured value.
gov.nist.javax.sip.TIMER_C	Configured value.
gov.nist.javax.sip.TIMER_D	Configured value.
gov.nist.javax.sip.TIMER_F	Configured value.
gov.nist.javax.sip.TIMER_H	Configured value.
gov.nist.javax.sip.TIMER_J	Configured value.

3.8 Statistics calculation

CM collects the statistics specified in [1]. **CM** calculates the statistics based on the events it generates. The table below indicates how statistics is affected due to events.

Table 61 Statistics calculation depending on events

Event	Updated statistics
Statistics event	• *Current calls* for call type unknown are incremented by one.
Failed event	• *Abandoned Rejected calls* are incremented by one if the Failed event indicates that the call is far or near end abandoned. Otherwise *Rejected calls* are incremented by one. • *Current calls* for call type unknown are decremented by one.

Event	Updated statistics
Error event	• If the call is not already connected or disconnected (i.e. not in states *Connected*, *Failed*, *Error*, or *Disconnected*) the *Current calls* for call type unknown are decremented by one and the *Rejected calls* are incremented by one. • If the call is connected but not already disconnected (i.e. not in states *Failed*, *Error*, or *Disconnected*) the *Current calls* for the call type voice or video are decremented by one and *Near End Disconnected calls* are incremented by one. • *Errors* are incremented by one.
Disconnected event	• If the call is not already disconnected (i.e. not in states *Failed*, *Error*, or *Disconnected*) one of *Far End Disconnected calls*, *Near End Disconnected calls*, and *Abandoned calls* is incremented by one depending upon the reason for the failed event. • If the call has been connected, *Current calls* for the call type voice or video are decremented by one. Otherwise the Current calls for call type unknown are decremented by one.
Connected event	• *Current calls* for the call type unknown are decremented. • *Current calls* for the call type voice or video are incremented by one. • *Connected calls* are incremented by one.
Dropped Packets event (this is a **CM** internal event generated when the inbound media stream is deleted)	• *Dropped packets* are incremented with the amount of dropped packets for the inbound stream during the entire call duration.

When the new call is initiated, the call type is not yet determined. Therefore a *Statistics event* is generated so the counter for current calls can be increased. When the call type is determined, a *Connected event* is generated. At this time the statistics is updated with the call type, i.e. the counter for unknown call type is decreased and the counter for voice or video calls is increased.

3.9 Threads and event queues

Currently **CM** uses a thread pool to receive threads in the following situations:

- One for each event queue type; i.e. one for the event queue in the controller, one for the event queue in the calls and one for the event queue in the SSP instances.

- One for global event handling. (Currently only the *Configuration Changed* event is handled.)
- One for statistics calculation.

An event queue guarantees that the events on the queue are handled in the same order they arrive and that one event is handled at a time (i.e. only one thread at a time handles events from the queue).

3.10 Configuration

Each time the configuration is loaded (at initialization and reload config) a new configuration container is created and the configuration variables are stored in it. When **CM** uses configuration values, it first gets the configuration container from the configuration reader and then reads the values from the configuration container. There are three reasons for this implementation:

- When reading configuration values, there is no need to handle exceptions issued from the configuration component. They are already handled by the configuration reader when it initializes the configuration container.
- To be able to hold two separate views of the configuration when the configuration changed. Active calls keep the old configuration container while new calls get a new.
- The configuration container has setters for all configuration values, so when writing test cases, it is possible to modify some configuration values at runtime. This cannot be done if values are always read from configuration component classes.

3.11 Reliable provisional responses

3.11.1 Inbound calls

Whether or not provisional responses shall be sent reliably depends upon the content of a received SIP INVITE request in combination with configuration.

If an inbound SIP INVITE request indicates that the extension "100rel" is required (using the *Require* header field), **CM** sends all non-100 provisional responses reliably.

If an inbound INVITE indicates that the extension "100rel" is supported but not required (using the *Supported* header field), **CM** looks at a configuration parameter in order to determine whether to send non-100 provisional responses reliably or not. The configuration can have three values: send all unreliably, send all reliably or to send only those carrying SDP information reliably.

3.11.2 Outbound calls

It is the remote party that decides if a provisional response shall be sent reliably or not for a SIP INVITE request initiated by **CM**. In the outbound SIP INVITE, CM indicates that the extension "100rel" is supported (using the *Supported* header field) but not required. When **CM** receives a provisional response sent reliably, a PRACK request is sent to acknowledge the response.

4 Third-Party Products and external interfaces

4.1 Third-Party Products and Freeware

Product name	Product version	Company	License	Delivered with the component	ECCN US/EU	Product No. and R-state
XML Beans	2.1.0	Apache Software Foundation.	Apache	Yes	EAR99/ 0	SWF0033 R2A
Used for generation and parsing of media control XML documents in SIP messages.						
NIST-SIP	1.2	National Institute of Standards and Technology	Public domain (Not copyrighted)	Yes	EAR99/ 0	SWF0054 R1A
SIP stack.						

4.2 External Products

Product Name	Company	Used for	Version of the integration tested product

4.3 External Protocols

Product Name	Specification	Used for	Version of the protocol

5 References

[1] FS – Call Manager
8/FS-MAS0001

[2] IWD SIP
7/IWD-1/HDB 101 02

[3] NIST SIP
http://www-x.antd.nist.gov/proj/iptel/

[4] JAIN SIP API (JSR 32)
http://www.jcp.org/en/jsr/detail?id=32

[5] JAIN SDP API (JSR 141)
http://www.jcp.org/en/jsr/detail?id=141

[6] SIP: Session Initiation Protocol
RFC 3261
http://ietf.org/rfc/rfc3261.txt?number=3261

[7] CCXML 1.0 specification (working draft)
http://www.w3.org/TR/CCXML/
(as of 2005-08-23)

6 Terminology

CCXML	Call Control XML
CM	Call Manager component
FSA	Finite State Automaton
JIT	Just-in-time compilation
JVM	Java Virtual Machine
OMM	Operate and Maintain Manager
SDP	Session Description Protocol
SIP	Session Initiation Protocol
VFU	Video Fast Update