Author	C. Arnault
Date	1999/12/15
Version	0.1
Document No.	VIR-MAN-LAL-5600-106

VIRGO - European Gravitational Observatory
Traversa H di Via Macerata - Santo Stefano a Macerata, -56021 Cascina, Italia.
Secretariat: Telephone (+39) 050 752 521 FAX (+39) 050 752 550

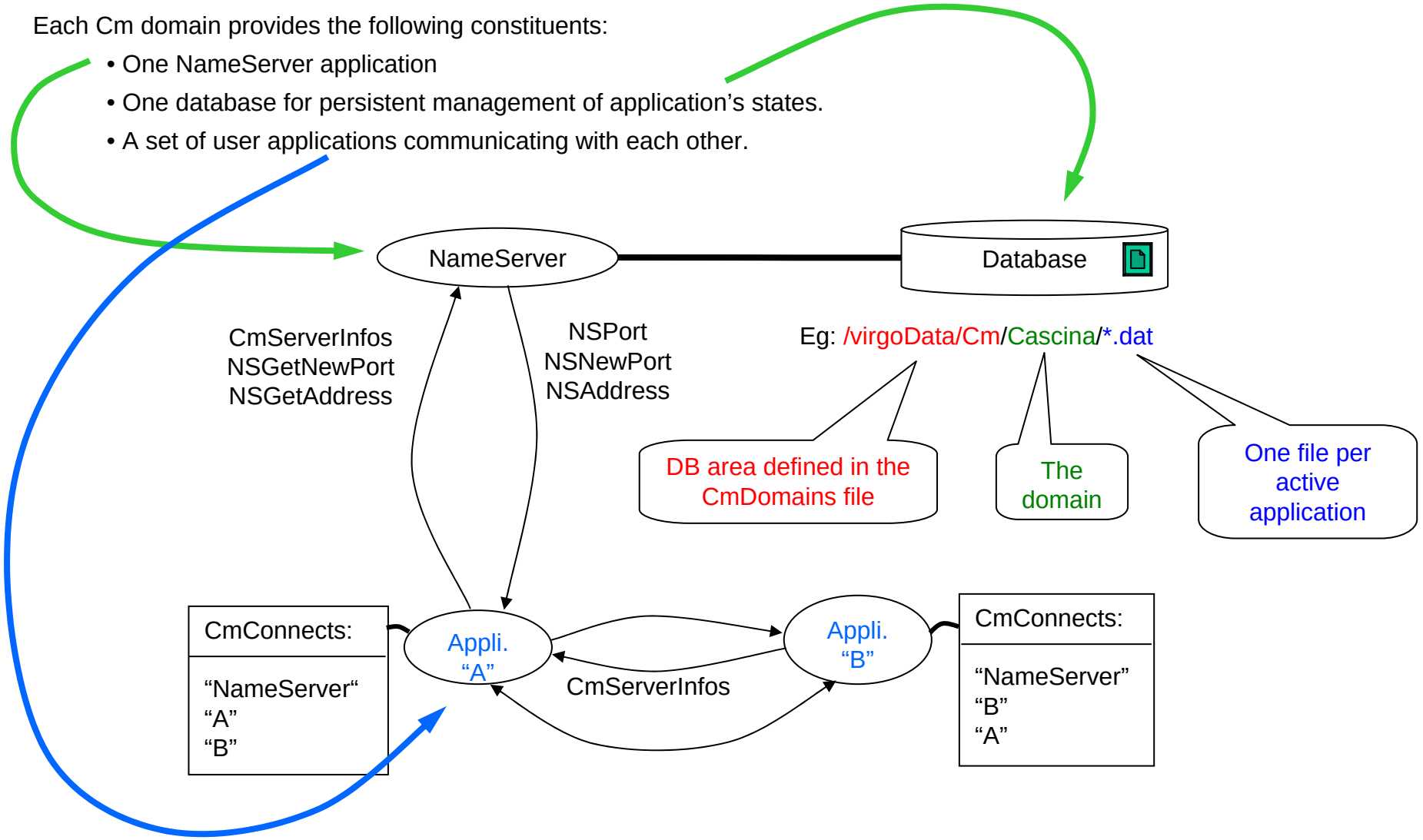
Cm design documentation

- General architecture for one Cm domain
- The internal Cm protocol
- The internal Cm protocol (continued)
- The NameServer database
- Operations during the CmMessageOpen(Multiple)Server
- Setting up the connection to the NameServer
- The NS port handler
- Operations during the CmMessageSend
- Operations during the CmMessagePost
- Operations during the CmConnectNew
- Operations during the CmConnectNewWithAddress
- Operations during the CmMessageWait
- Operations during the CmMessageCheck
- Internal structure of a CmMessage object

General architecture for one Cm domain

Each Cm domain provides the following constituents:

- One NameServer application
- One database for persistent management of application's states.
- A set of user applications communicating with each other.



The internal Cm protocol 

This section describes the Cm messages internally used by the Cm package, listed by their types, either between applications and the NameServer or between applications

CmServerInfos

Any application willing to establish a connection towards another application first sends this message to it. No answer is expected from it except for the NameServer which returns a NSPort message.

```
string protocol_version
string name
string host_name
int    port_number
int    is_multiple
string owner
```

NSPort

The NameServer just accepted a connection from a new application (using CmMessageOpenServer or CmMessageOpenMultipleServer). It allocates a new port number and sends it back to the application. In addition, and only for *multiple* servers, the effective application name (including the suffix _<n> will be sent).

```
string original_name
int    allocated_port
string new_name
int    transaction_id
```

NSGetNewPort

An application just tried to bind to a port number (previously proposed by the NameServer). But this port number is not available. The application sends back this message to the NameServer so as to obtain another port number.

```
string connection_name
int    transaction_id
```

NSNewPort

The NameServer sends back to the application another port number after a bind failure.

```
string connection_name
int    new_port
int    transaction_id
```

The internal Cm protocol (continued)

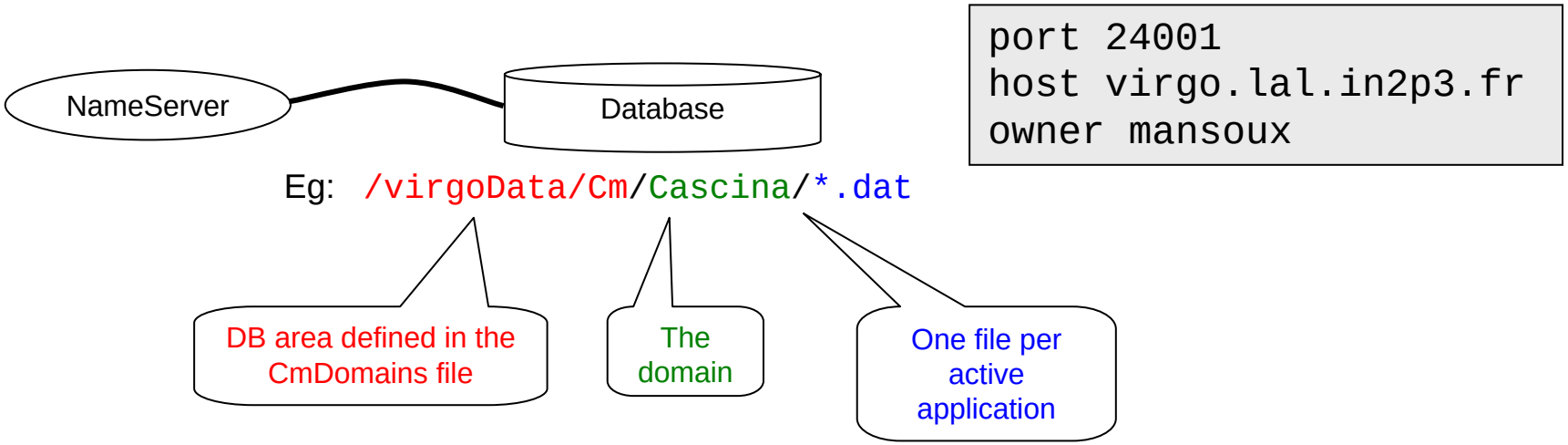
NSGetAddress	An application wishes to open a new connection towards another application. It first sends this message to the NameServer, which in turn will answer using the NSAddress message.	`string name`
		`int reserved`
		`int transaction_id`

NSAddress	The NameServer sends this message as an answer to the NSGetAddress message.	`string name`
		`string host_name`
		`int port_number`
		`int is_multiple`
		`string owner`
		`int transaction_id`

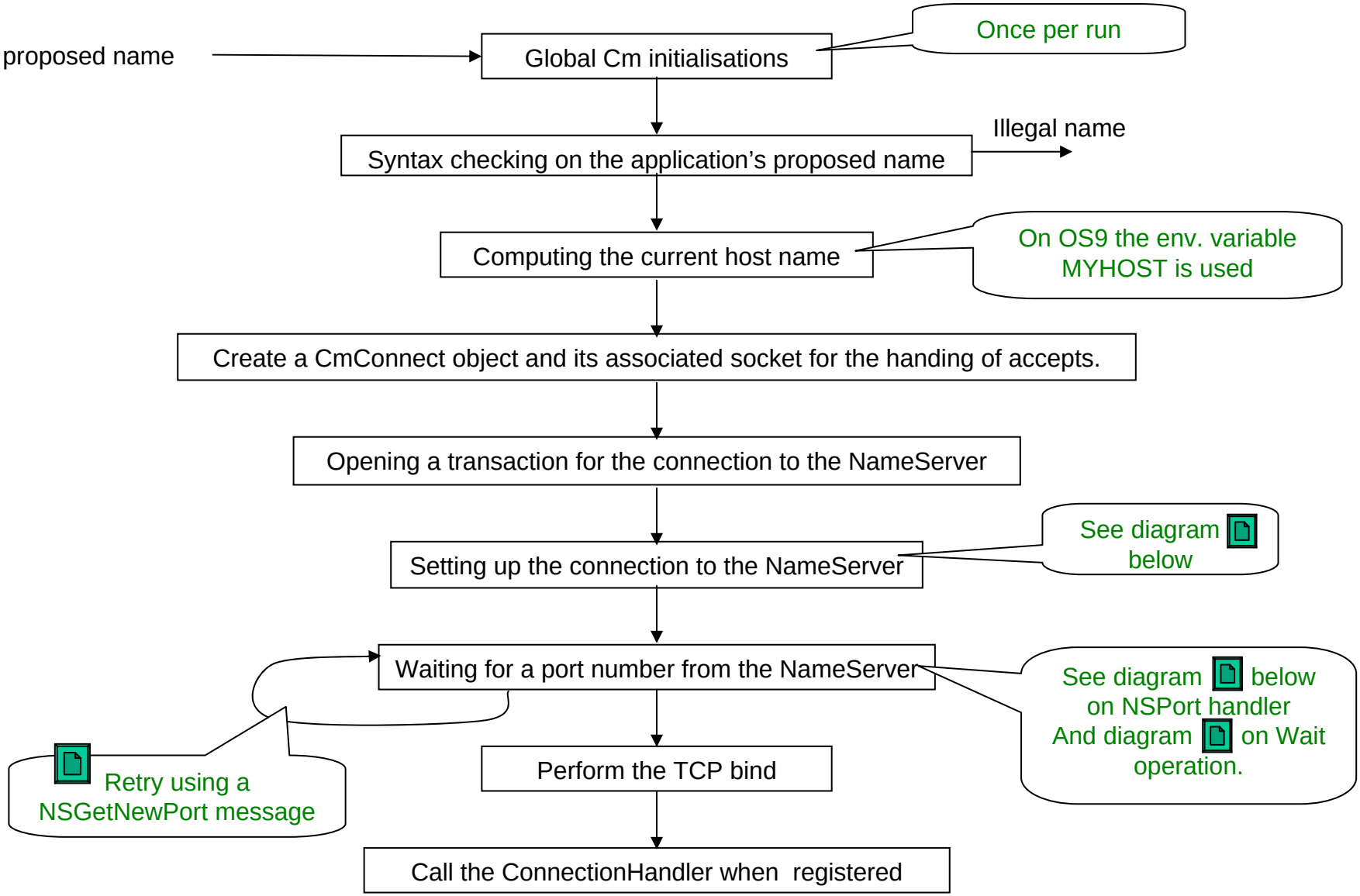
The NameServer database

- **The database files are named with the application's name and suffixed with `.dat`. They hold**
  - The internet address and the port number allocated to the application
  - The name of the user who launched the application

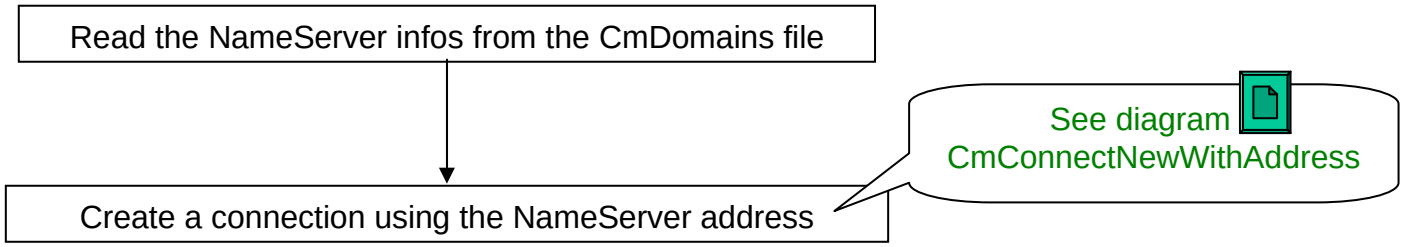
- **Each database file is renamed with the suffix `.bck` when the application dies.**



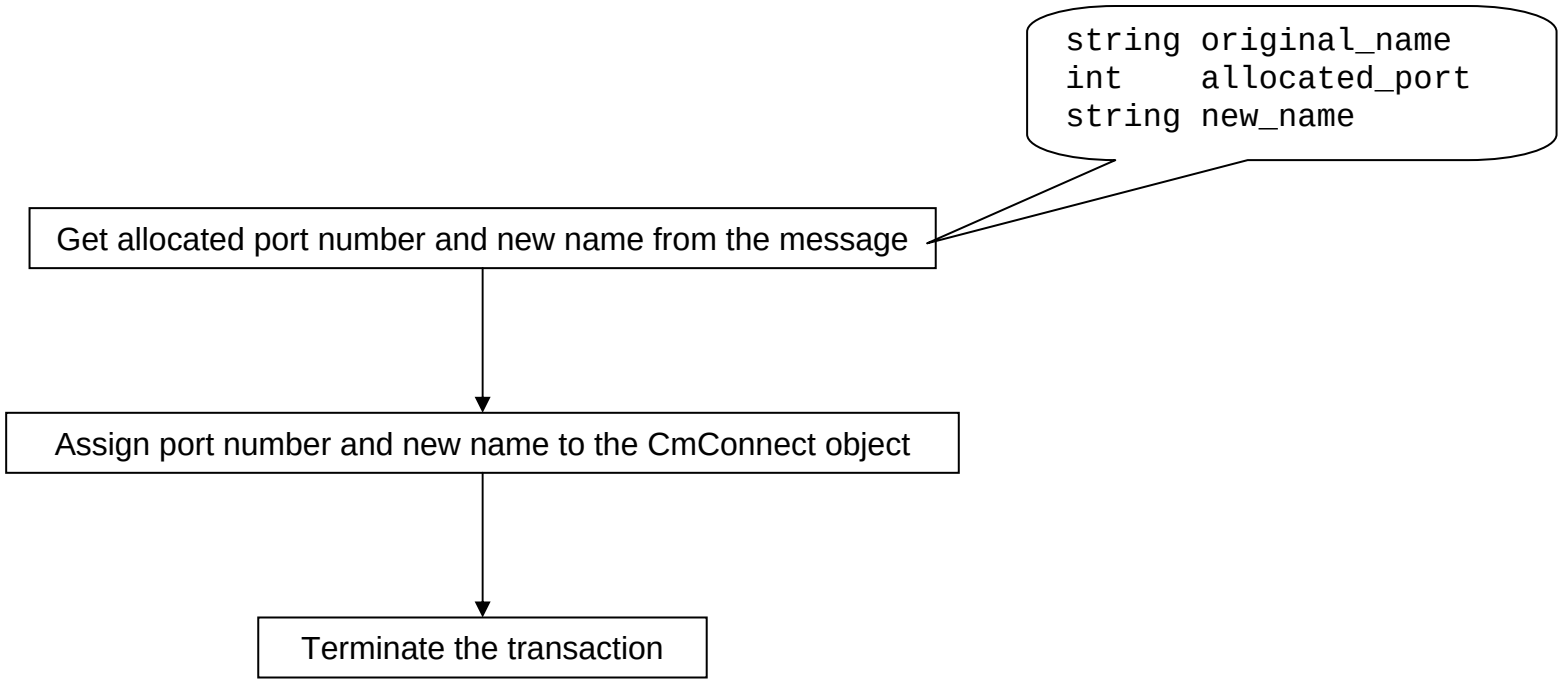
Operations during the CmMessageOpen(Multiple)Server



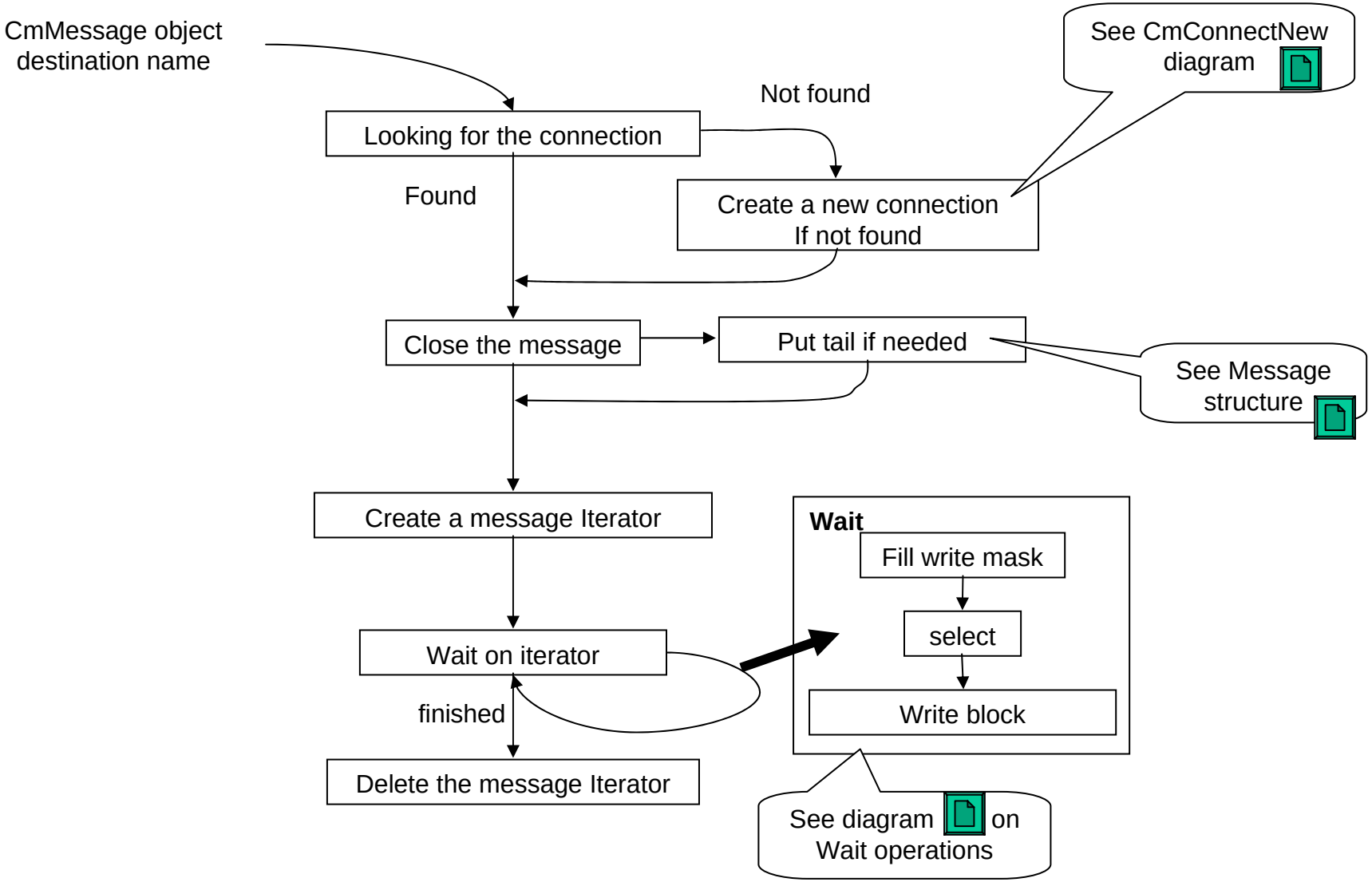
Setting up the connection to the NameServer



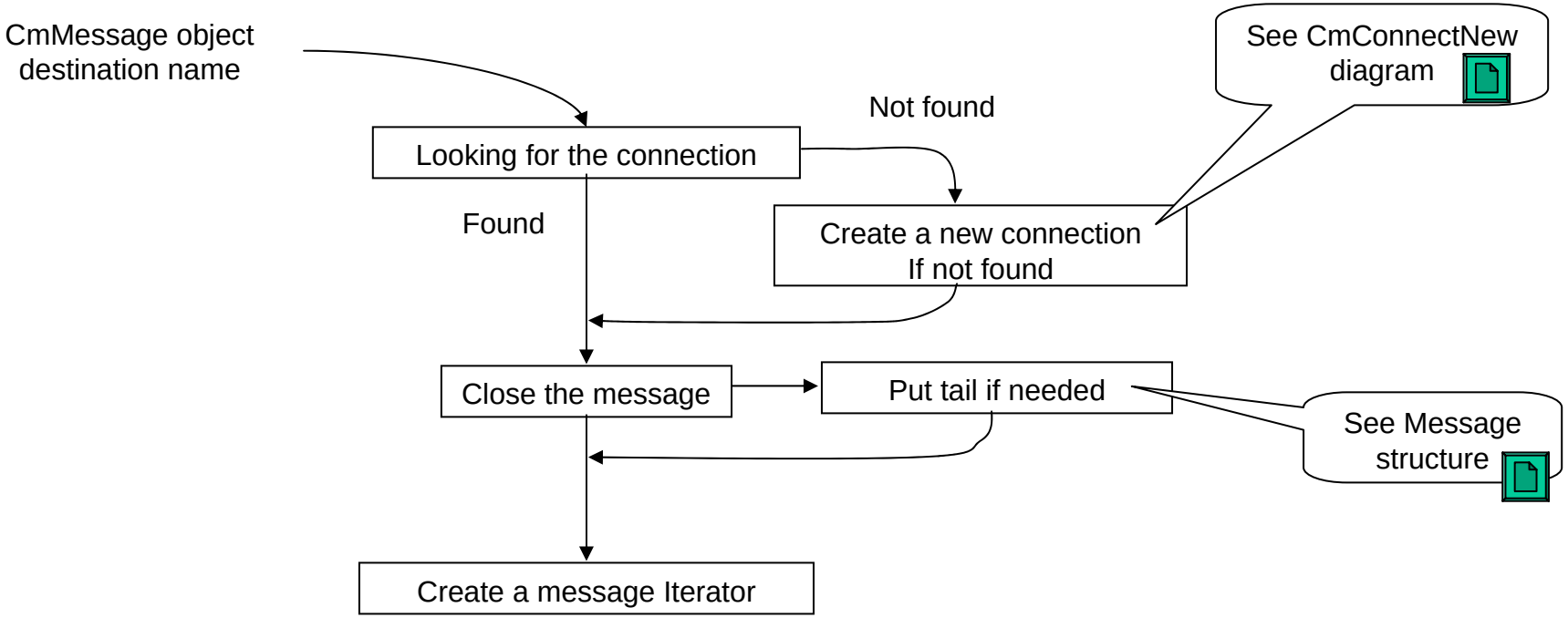
The NS port handler



Operations during the CmMessageSend



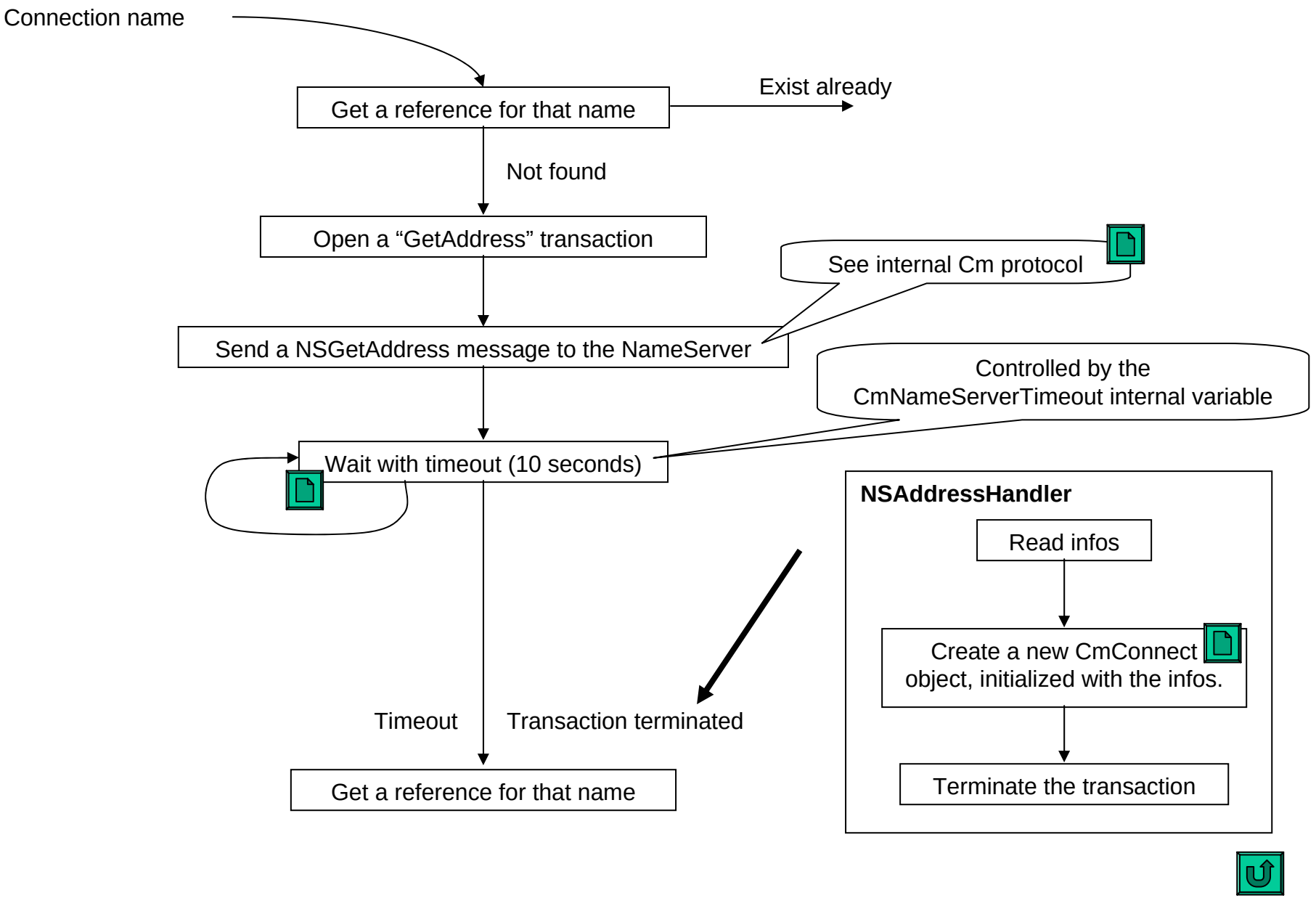
Operations during the CmMessagePost



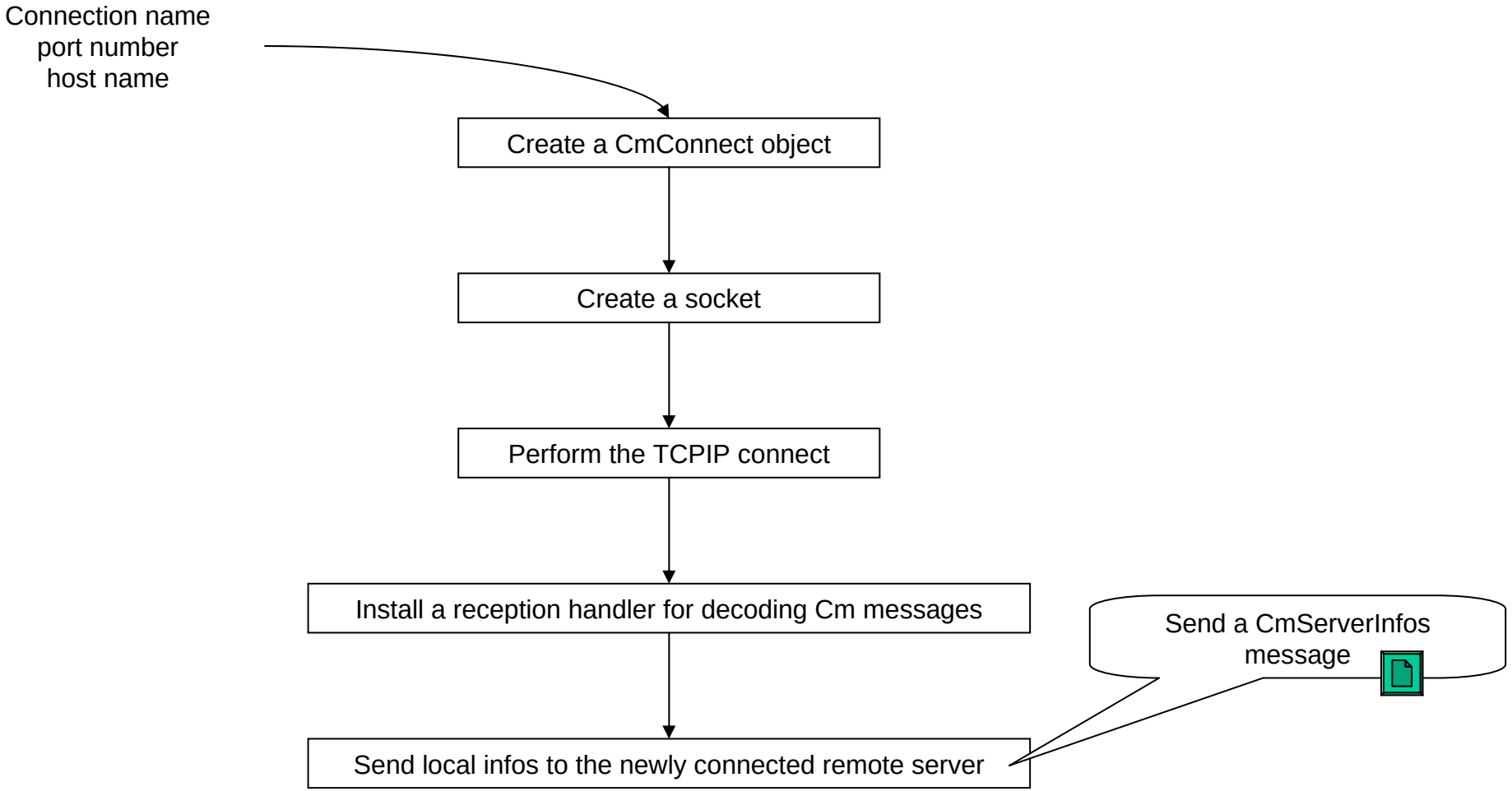
The iteration on sending partial message blocks will be performed at each subsequent call to either CmMessageWait or CmMessageCheck.

Eventually, when the send is completed, an optional handler will be called.

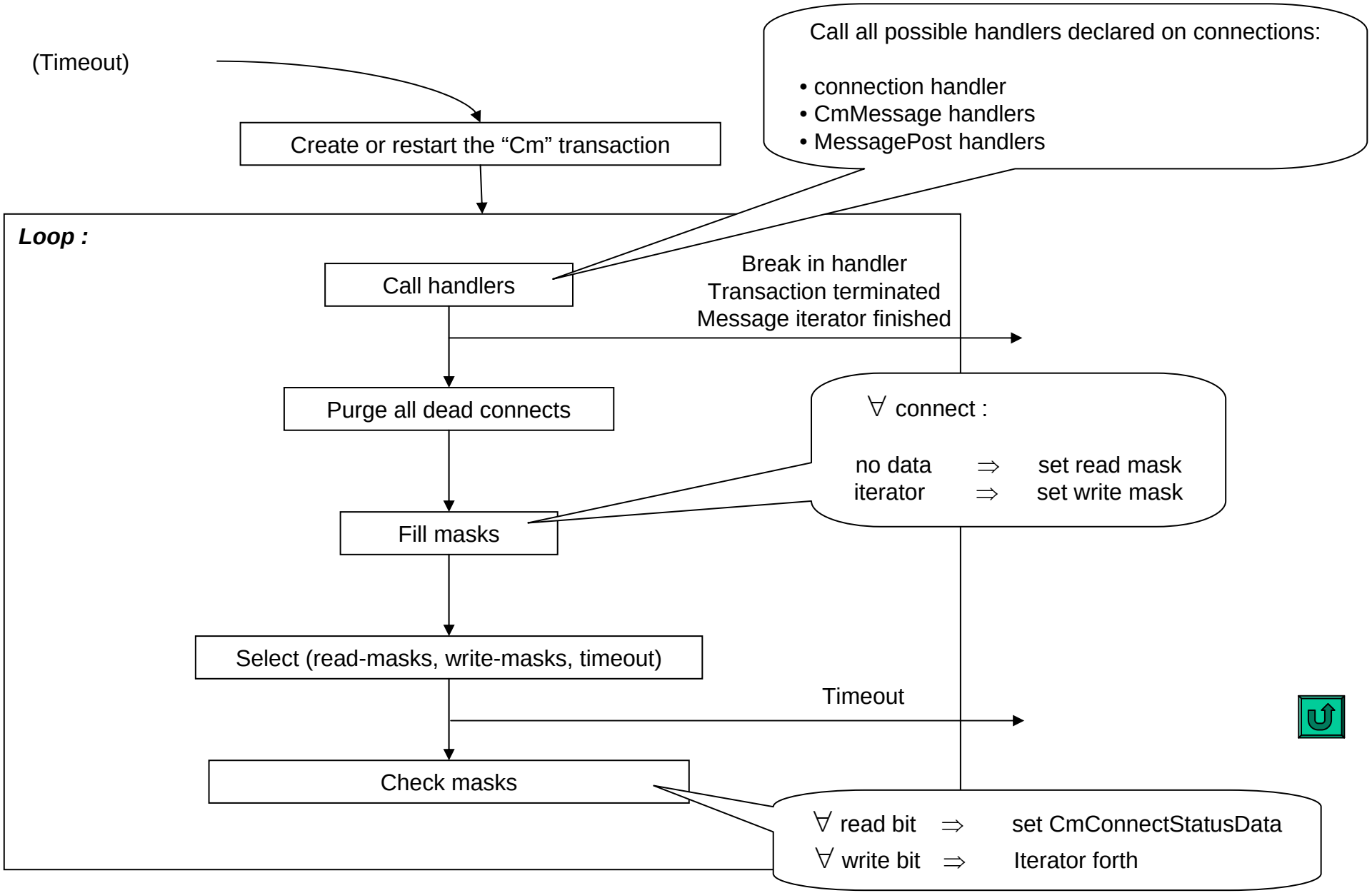
Operations during the CmConnectNew



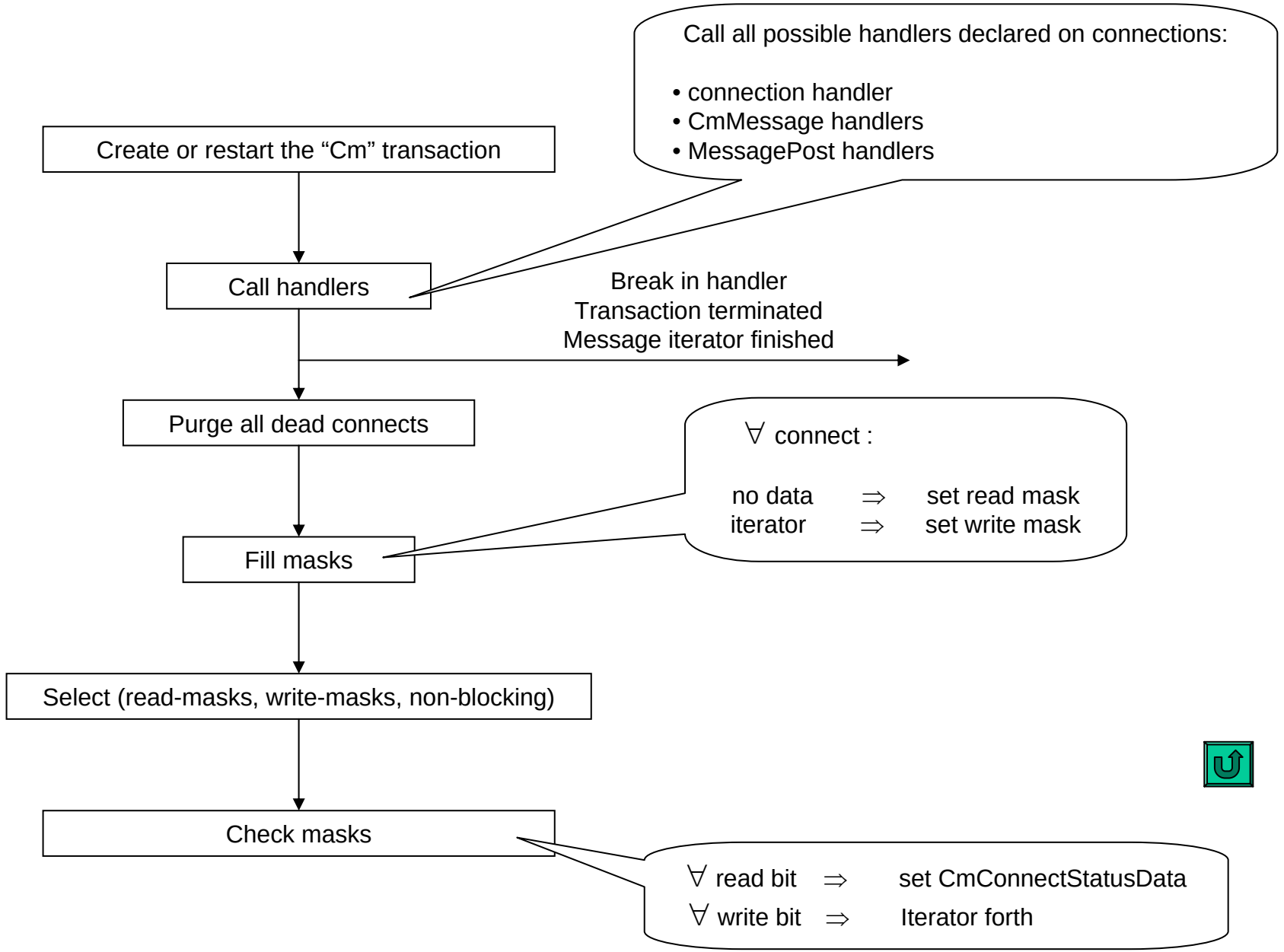
Operations during the CmConnectNewWithAddress



Operations during the CmMessageWait



Operations during the CmMessageCheck



Internal structure of a CmMessage object