# pContactMgrV20: Managing Platform Contacts

**August 2022**

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

`project-pavlab/appdocs/app_pcmanager_v20`

# 1 Overview

Note: The pContactMgrV20 app is a substantially re-written version of its predecessor, pBasicContactMgr. The new version is meant to be largely backward compatible with the previous version. The new version has better support for larger streams of contact information over longer missions. It has better guards on memory management and should be more efficient in terms of CPU load. It also has much better support for filtering contacts with greater control over how different contacts may be handled uniquely. The latter is in support of apps in the Swarm Autonomy Toolbox which require these features from a contact manager.

The pContactMgrV20 application deals with information about other known vehicles in its vicinity. It is not a sensor application, but rather handles incoming "contact reports" which may represent information received by the vehicle over a communications link, or may be the result of on-board sensor processing. The primary use case involves the generation of alert messages corresponding to contact relative position criteria configured by the consumers of the alerts. The pContactMgrV20 posts to the MOOSDB summary reports about known contacts, which supports functionality of other MOOS applications.

The basic idea is shown below:

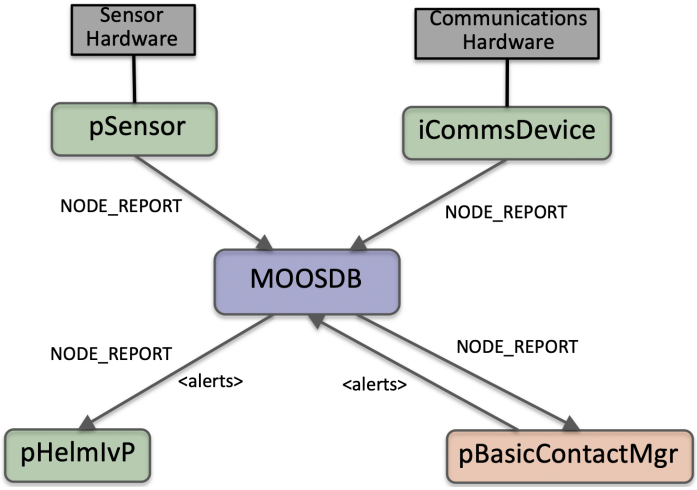


Figure 1: **The pContactMgrV20 Application:** The pContactMgrV20 utility receives NODE_REPORT information from other MOOS applications and manages a list of unique contact records. It may post additional user-configurable alerts to the MOOSDB based on the contact information and user-configurable conditions. The source of contact information may be external (via communications) or internal (via on-board sensor processing). The pSensor and iCommsDevice modules shown here are fictional applications meant to convey these two sources of information abstractly.

The contact manager is typically used in coordinating with the IvP Helm, but a key design goal of the contact manager is to remain helm agnostic. The design or configuration of the contact manager need not know anything about the helm or any other application for that matter. The configuration or design of the helm, and helm behaviors, on the other hand needs to coordinate its function with the contact manager. Although contact manager alerts are simply MOOS variables, with no explicit

linking or dependency on the helm or helm libraries, the alerts are commonly used by the helm to trigger the spawning of a behavior associated with a contact (Section 4). The configuration of the contact manager, for use with the helm, is also the job of the helm behaviors which post a MOOS variable to request dynamic alert configurations of the contact manager (Section 4.1).

# 2 Contact Alert Structure and Configuration

A *contact alert* is an internal data structure of the contact manager, which may be configured to comprise several types of alerts in a given mission. Configuration of alerts may be done through the mission (`.moos`) file, or they may be configured by other apps in the MOOS community that publish alert request. The latter is the most common, but both may coexist. Here the data structure is defined, start-up configuration from the mission file, and run-time configuration through MOOS mail messages.

## 2.1 The Contact Alert Structure

The *contact alert* data structure contains the following fields:

The core components:

- `alert id`: A unique name for the alert.
- `alert_range`: A minimal range in meters that will cause an alert trigger.
- `cpa_range`: A larger range in meters that may trigger an alert before breaching the minimal range.
- `on_flag`: An alert posting, or `VarDataPair`, to be published when the alert has been triggered. Multiple flags may be provided.
- `off_flag`: An alert posting, or `VarDataPair`, to be published when the alert trigger criteria has switched off. Multiple flags may be provided.

The filter components:

- `ignore_name`: One or more contact names to be ignored.
- `match_name`: One or more contact names to be matched.
- `ignore_type`: One or more contact types to be ignored.
- `match_type`: One or more contact types to be matched.
- `ignore_group`: One or more contact groups to be ignored.
- `match_group`: One or more contact groups to be matched.
- `ignore_region`: A region where contacts will be ignored.
- `match_region`: A region where contacts will be ignored.

### Range Components

The `alert_range` component represents a threshold range to a contact, in meters. When a contact moves within this range, an alert will be generated. This is discussed in Section 3.1. The `cpa_range` component also represents a threshold range, in meters. Typically this `cpa_range` is greater than

the alert_range parameter. It is never less than the alert_range. When a contact is noted to be within the cpa_range, the contact and ownship trajectories are considered, to calculate the closest point of approach (CPA). If the CPA range is determined to be within the alert_range, an alert is generated, even if the present range between ownship and contact is outside the alert_range. This is also discussed in Section 3.1.

#### Flag Posting Components

A flag describes a MOOS posting. It will be a pair, in the form of MOOS_VARIABLE = value. The value component may be string containing macros to allow the alert posting to contain information about the contact or ownship. This is discussed more in Section 2.3.

#### Filter Components

The filter components allow the user to require that a contact meet a configured set of criteria in order for the alert to be applicable. This criteria can be based on either the contact name, platform type, group name, or region of the operation area where the contact presently resides. The *ignore* options indicate the contact should be ignored if one or more of the criteria hold. The *match* options indicate the contact should be ignored *unless* the match options are satisfied. This is discussed further in Section 5.

#### Alert Validity

The alert ID, the alert range, and one or more postings are regarded as mandatory for the alert to be considered valid. All other fields are optional. The *cpa range* must be a value equal to or larger than the *alert range*. An invalid alert provided on startup will result in a configuration warning. An invalid alert received via an incoming mail message will result in a run warning.

### 2.2 Alert Configuration from the Mission File

Alerts may be configured in the mission file with the alert parameter. A single alert type may be defined over several lines, where each line contains the alert id of the alert type being configured. For example:

```
alert = id=<alert-id>, on_flag=<VarDataPair>, off_flag=<VarDataPair>
alert = id=<alert-id>, alert_range=<distance>,  cpa_range=<distance>
```

A simple example:

```
alert = id=K8, on_flag=SAY_MOOS=alarm, off_flag=SAY_MOOS=ok, alert_range=20, cpa_range=30
```

Or, equivalently:

```
alert = id=K8, on_flag=SAY_MOOS=alarm
alert = id=K8, off_flag=SAY_MOOS=ok
alert = id=K8, alert_range=20
alert = id=K8, cpa_range=30
```

### 2.3 Alert Configuration from Incoming Mail

Alert configuration can also be requested at run time, originating from another MOOS app via a message to BCM_ALERT_REQUEST. The contents of this message are identical to the one-line configuration used in a mission file shown above.

```
BCM_ALERT_REQUEST = id=k8, on_flag=SAY_MOOS=alert, alert_range=20, cpa_range=40
```

This kind of dynamic request is essential to the operation of the IvP Helm for contact related behaviors such as collision avoidance. This is discussed in Section 4.

### 2.4 Flag Macros

The on_flag component of an alert is a posting, or VarDataPair, that will be published by pContactMgrV20 to the MOOSDB when an alert is triggered. The data component of this posting may be a string, and may contain one or more supported macros to be expanded at the time of the post. For example the $[VNAME] macro contains the name of the contact related to the alert. This can be used to spawn a collision avoidance behavior as discussed in Section 4. The full set of supported macros is below:

| | |
|---|---|
| $[VNAME]: | The name of the contact. |
| $[X]: | The position of the contact in local $x$ coordinates. |
| $[Y]: | The position of the contact in local $y$ coordinates. |
| $[LAT]: | The latitude position of the contact in earth coordinates. |
| $[LON]: | The longitude position of the contact in earth coordinates. |
| $[HDG]: | The reported heading of the contact. |
| $[SPD]: | The reported speed of the contact. |
| $[DEP]: | The reported depth of the contact. |
| $[VTYPE]: | The reported vessel type of the contact. |
| $[UTIME]: | The UTC time of the last report for the contact. |

Multiple macro types may be used in the same on_flag or off_flag.

# 3 Alert Triggering

## 3.1 Trigger Criteria Base on Range and CPA Range

Alerts are triggered for all contacts based on range between ownship and the reported contact position. It is assumed that each incoming contact report minimally contains the contact's name and present position. An alert will be triggered if the current range to the contact falls within the distance given by `alert_range`, as in Contact-A in Figure 2.

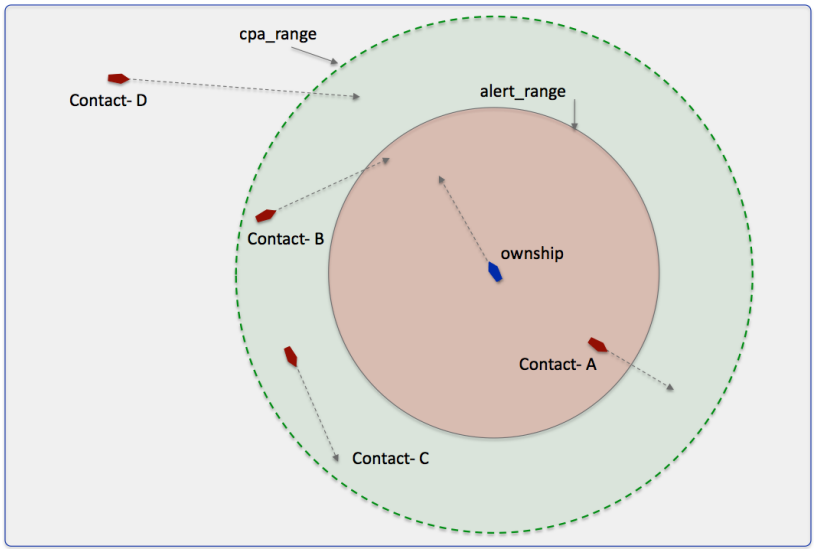


Figure 2: **Alert Triggers:** An alert may be triggered by the contact manager if the contact is within the `alert_range`, as with Contact-A. It may also be triggered if the contact is within the `cpa_range`, and the contact's CPA distance is within the `alert_range`, as with Contact-B. Contact-C shown here would not trigger an alert since its CPA distance is its current range and is not within the `alert_range`. Contact-D also would not trigger an alert despite the fact that its CPA with ownship is apparently small, since its current absolute range is greater than `cpa_range`.

An alert may also be configured with a second trigger criteria consisting of another range to contact, `cpa_range`. The range is enforced to be equal to or larger than the `alert_range`. If `cpa_range` is left unspecified, it will be set to the `alert_range`, effectively disabling this feature. When a contact is outside the `alert_range`, but within the `cpa_range`, as with Contact-B in Figure 2, the closest point of approach (CPA) between the contact and ownship is calculated given their presently-known position and trajectories. If the CPA distance falls below the `alert_range` value, an alert is triggered.

### Turning Off an Alert and a Schmitt Trigger

Once an alert has been triggered, presumably a posting has been made, specified with the `on_flag` parameter. Typically there is nothing further to do when the trigger criteria is no longer met. As discussed later in the case of the helm, a collision avoidance behavior will spawn with initial alert, and the behavior will be deleted later when the contact passes beyond the `cpa_range` of ownship. The only catch is that the same contact, with the same contact ID, may later change course and once again close range on ownship. A new alert will need to be triggered, and new behavior spawned.

For this reason, when the contact opens beyond the `cpa_range`, the contact manager marks the contact/alert pair as *off*. If that contact does indeed change course and approach ownship, a new alert will be trigger, a new posting will be published and a new collision avoidance behavior will be spawned. However, when events are triggered on/off based on a range threshold, it opens the door that a noisy sensor may create a thrashing of a rapid succession of alert on/off declarations and behavior spawning and deletions. In physics there is the notion of a *Schmitt trigger* which is a switch with a small positive feedback that increases the threshold for changing state in both directions. For example this could be accomplished in our situation by adding a few meters to the `cpa_range` as the contact is opening range, and subtracting a few meters from `cpa_range` when the contact is closing range to ownship. This isn't necessary however since the nature of the `cpa_range` is such that, for a contact to trigger an alert, it must not only be closing range to ownship, it must be on a heading that creates a CPA to ownship within the `range` setting (like Contact-B in Figure 2). As a contact is opening range, and turning off the alert, even if it were magically within the `cpa_range` an instant later, it would not have a CPA within the `range` circle until it comes about with a significant heading change. This creates natural Schmitt trigger effect.

### 3.2 Alert Triggering Internal Data Structures

The contact manager maintains a few key data structures for managing alert generation and other status updates:

- Contact Status Table
- Contact Alert Table
- Retired Contacts

As contact information is updated in the contact manager, the contact's position and timestamp of the most recent update is stored in the contact status table. This range is the simple linear distance. The CPA range projects both vehicles into the future to calculate the expected CPA range. The extrapolated range is based on an extrapolated contact position based on the current time and time of the last position update for that contact.

Each data structure is depicted below:

Contact Status

| | | | | |
|---|---|---|---|---|
| Contact = C134 | Position/TimeStamp | range | cpa range | extrapolated range |
| Contact = C008 | Position/TimeStamp | range | cpa range | extrapolated range |
| Contact = C344 | Position/TimeStamp | range | cpa range | extrapolated range |
| Contact = C083 | Position/TimeStamp | range | cpa range | extrapolated range |

Alert Status

| | Alert ID=K34 | Alert ID=K07 | Alert ID = K04 |
|---|---|---|---|
| Contact = C134 | **on** | off | **on** |
| Contact = C008 | off | off | off |
| Contact = C344 | **true** | **on** | off |
| Contact = C083 | off | off | off |

Retirement List

| | |
|---|---|
| Retired Contacts | C091, C002, C023, C430, C055, C493, C588, C008, C331, C692 |

Figure 3: **Internal Data Structures:** Information is updated based on the arrival new contact information, updates on internally calculated status, and retiring of contacts out of range and not recently updated.

Based on the updated values of the contact status table, each contact is judged based on the alert range criteria for each alert. The contact alert table is updated accordingly. If a cell switches from `false` to `true`, the `on_flag(s)` for that alert and contact are posted. If the cell switches from `true` to `false`, the `off_flag(s)` are posted.

A contact may be *retired* for one of two reasons. (1) It may have been a long time since its last position update, longer than the threshold set by `contact_max_age`, which has a default of 600 seconds. (2) Or it may now be far away, beyond the range set by the parameter `reject_range` which has a default value of 2000 meters. If it is retired, the contact is reclaimed from the two tables, and is placed on the retirement list. This list also has a maximum size, pruned first-in-first-out.

### 3.3 Publications about Internal Status

There are several MOOS variables published by the contact manager that reflect some of the information in the above internal data structures:

- `CONTACTS_LIST`: A comma-separated list of names for all contacts in the contact status table.
- `CONTACTS_ALERTED`: A list of contacts that are currently in the alerted state, per alert ID.
- `CONTACTS_COUNT`: The number of currently alerted contacts.
- `CONTACTS_UNALERTED`: A list of contacts that are currently in the unalerted state, per alert ID.
- `CONTACTS_RETIRED`: A comma-separated list of names for retired contacts.
- `CONTACTS_RECAP`: A summary, name, range and report age, for contacts in the contact status table.

Here are some examples:

```
CONTACTS_LIST      = delta,gus,charlie,henry
CONTACTS_ALERTED   = (delta,avd)(charlie,avd)
```

```
CONTACTS_COUNT     = 2
CONTACTS_UNALERTED = (gus,avd)(henry,avd)
CONTACTS_RETIRED   = bravo,foxtrot,kilroy
CONTACTS_RECAP     = name=ike,age=11.3,range=193.1 # name=gus,age=0.7,range=48.2 #
                     name=charlie,age=1.9,range=73.1 # name=henry,age=4.0,range=18.2
```

All variables are only published when changed. The CONTACTS_RECAP variable will very likely change content on each iteration since it contains the numerical ranges to moving contacts from a moving ownship. If published and logged on every iteration this will generate large amounts of perhaps unnecessary log data. By default, this variable will publish once per second, but this is changeable with the contacts_recap_interval. It may be set to any non-negative value, or the string "off" to turn it off completely. Of course if log file size is a concern, the frequency of logging can always be adjusted in the pLogger settings.

A fair question might be - why retain in memory any information related to contacts that currently do not have an active alert. As will be discussed in Section 6.1 and Section 6.2, the contact manager generates other reports that may include contacts that may not presently warrant an alert.

# 4 Contact Manager Coordination with the Helm

The contact manager was designed primarily to support the IvP Helm, but also designed in a manner to make as few assumptions as possible about the helm implementation. The helm is a contact manager user, a consumer of contact manager alerts. It is the helm's responsibility to submit registrations to the contact manager for alerts, with the alert registration defining the conditions and format for the desired alert. The contact manager will blindly abide, and will generate alerts of the requested format, when the requested range conditions and filter conditions (if any) are met. This process is described in more detail in this section. More details on helm behavior templates, updates and behavior spawning can be found in the helm documentation.

## 4.1 Helm Registration for Alerts

When the helm starts up, it populates a set of behaviors. Some behaviors are static, created at launch time, and persist throughout the mission. Other behaviors are configured as templates, placeholders for perhaps many future behaviors of that type. This is the case for contact-related behaviors such as collision avoidance. As Figure 4 shows below, the behavior template is implemented to post an alert registration message upon startup.

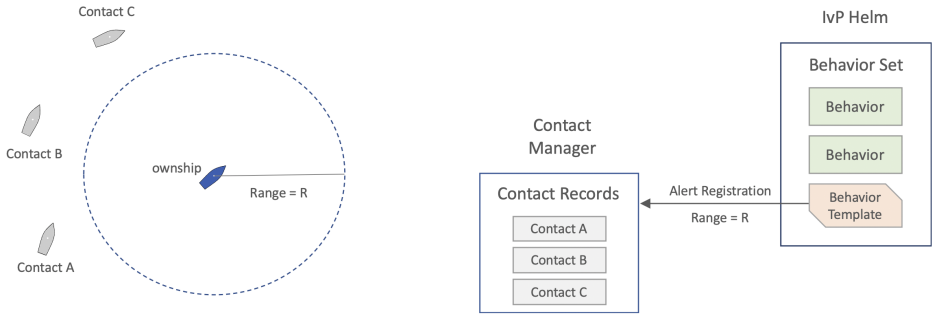


Figure 4: **Alert Registration**: When the helm starts, contact-related behaviors, configured as templates, will post a registration to the contact manager. The registration contains information about the desired format and threshold criteria for future alerts.

The alert registration format was discussed previously in Section 2.3. For helm contact-related behaviors, the `on_flag` specifies a posting to the MOOS variable named in the behavior's `updates` parameter configuration, e.g., `CONTACT_INFO` in the below example.

```
BCM_ALERT_REQUEST = id=avoid, alert_range=20, cpa_range=40, ignore_type=cargo \
                    on_flag=CONTACT_INFO=name=avd_$[VNAME] # contact=$[VNAME]
```

The rest of the alert request specifies the alert and cpa range values so the contact manager knows the conditions this behavior template is expecting to receive alert postings, i.e., publications to `CONTACT_INFO` in this case. After this start up process, the helm, with respect to this behavior is in listen-and-wait mode, until an alert is received.

## 4.2 Helm Action Upon Receiving an Alert

Figure 5 below depicts alert sequence of events. First a contact crosses a range threshold relative to ownship. Second, this is detected and noted in the contact manager. This results in an alert posting of a format that a behavior template in the helm is expecting. Finally the helm will spawn a behavior dedicated exclusively to dealing with this contact.

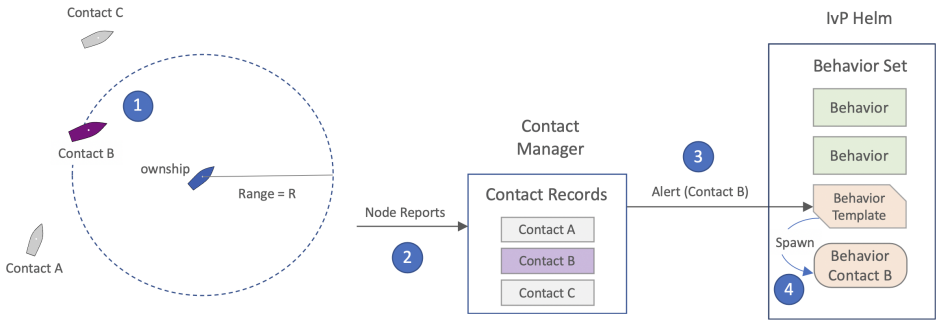


Figure 5: **Alert to Spawning**: (1) A contact closes range to ownship, crossing a range threshold. (2) The contact manager makes note and changes the internal record associated with the contact. (3) An alert it generated and received by the helm. (4) The helm spawns a new contact-related behavior dedicated to this contact.

The posting, following the `BCM_ALERT_REQUEST` example above, will have the following form:

```
CONTACT_INFO=name=avd_$[VNAME] # contact=$[VNAME]
```

The contact manager will use the contact name, e.g., `vessel_23`, to expand the `$[VNAME]` macros and the posting will look more like:

```
CONTACT_INFO=name=avd_vessel_023 # contact=vessel_023
```

See the helm documentation for more information on behavior updates and template behaviors. In short, the helm will first check to see if a behavior already exists that (1) is listening for parameter updates through the MOOS variable `CONTACT_INFO`, and (2) has the unique name of `avd_vessel_23`. If so, it would regard the remainder of the `CONTACT_INFO` message as an update to the existing behavior. If a behavior by that name does not already exist, and there is a behavior template that is listening for parameter updates on the MOOS variable `CONTACT_INFO`, a new behavior is then spawned with the name `avd_vessel_023`, and the remainder of the `CONTACT_INFO` message is treated as an update to this newly spawned behavior.

## 4.3 Helm Action Upon Alert Retirement

When a contact opens range to ownship sufficiently far, beyond the `cpa_range`, the contact manager will alter its internal state associated with the contact/alert pair. See the alert status table in Figure 3. From the helm's perspective no further action is required. Contact-related behaviors are implemented to *complete*, and be promptly deleted by the helm, once the contact goes beyond a

certain range. As you might guess, this range is automatically set to be equal to the `cpa_range` the contact manager uses. So, roughly at the same time, the contact manager denotes the alert for this contact to be *off*, and the helm lets the contact behavior complete and be removed from memory. If the contact and ownship should later cross paths again, the contact manager would simply generate a new alert and the helm would spawn a new behavior.

An example mission is available for showing the use of the contact manager and its coordination with the helm to spawn behaviors for collision avoidance. This mission is `m2_berta` and is described in the IvP Helm documentation. In this mission two vehicles are configured to repeatedly go in and out of collision avoidance range, and the contact manager repeatedly posts alerts that result in the spawning of a collision avoidance behavior in the helm. Each time the vehicle goes out of range, the behavior completes and dies off from the helm and is declared to the contact manager to be resolved.

# 5 Exclusion Filters

In certain missions it is desirable to treat different contacts differently. This may be based on the contact platform type, e.g., sailboats versus motorboats. It may be based on contact group affiliation, e.g., red team versus blue team. It may be based on contact location in the operation area, or perhaps simply relative to ownship. One example that comes up frequently in field trials is the desire for the robot to ignore the nearby test boat holding the test operators.

There are a number of options for configuring the contact manager to achieve the desired effect. In all cases, the configuration may be done at the alert level, or at the contact manager level. At the contact manager level, one may simply want to ignore all sailboats. At the alert level it's also possible that the helm may have one collision avoidance behavior configured and tuned for motorboats and should ignore sailboats, and another behavior configured and tuned for sailboats and should ignore motorboats.

Filters are also generally of a *match* type or an *ignore* type. For example the contact manager may be configured to ignore all sailboats by specifying an ignore filter based on sailboats. Or it may be configured to only regard sailboats by setting a match filter for sailboats. If multiple ignore filters are provided, a contact will be ignored if any one ignore filter applies to the contact. If multiple match filters are provided, a contact will be ignored only if all match filters do not apply to the contact. If any criteria fail, the contact will be ignored. For example, for a given sailboat, if sailboats are explicitly named in a match filter, but also named in an ignore filter, it will be ignored.

## 5.1 Exclusion Filters Based on Contact Name, Type or Group

Name, type and group filters are available at the contact manager level with the following pContactMgrV20 configuration parameters:

- match_name: A comma-separated list of match names
- match_group: A comma-separated list of match groups
- match_type: A comma-separated list of match types
- ignore_name: A comma-separated list of names to ignore

- ignore_group: A comma-separated list of groups to ignore
- ignore_type: A comma-separated list of types to ignore

Here are some examples:

```
match_name   =  henry
match_group  =  blue_team
match_type   =  motor_boat, crew_shell
ignore_name  =  abe,ben, cal
ignore_group =  red_team, work_boat
ignore_type  =  sail_boat
```

If a contact survives the filters at the contact manager level, it still must survive any filters that may have been declared at the individual alert level. The `match_group`, `match_type`, `ignore_group`, and `ignore_type` components work the same, but are embedded in an alert configuration. As a pContactMgrV20 configuration parameter, it may look like:

```
  alert = id=K8, on_flag=SAY_MOOS=alarm, off_flag=SAY_MOOS=ok, ignore_name=hal   \
          alert_range=20, cpa_range=30, match_group=blue_team
```

As an alert configured by a mail request to the contact manager, it may look like:

```
BCM_ALERT_REQUEST = id=k8, onflag=SAY_MOOS=alert, alert_range=20, cpa_range=40, \
                    match_group=blue_team
```

If a contact does not report its type information, it may be rejected if (a) there is at least one ignore_type specified, *and*, (2) the contact manager configuration parameter strict_ignore is set to `true`, which is the default value. The same holds true if the contact has missing group information.

## 5.2 Exclusion Filters Based on Contact Position Relative to a Region

Like the group and type filters, region filters are available at the contact manager level with the following pContactMgrV20 configuration parameters:

- match_region: A convex polygon
- ignore_region: A convex polygon

Multiple regions of each kind may be specified. These regions may be contiguous, overlapping or non-overlapping. Thus non-convex regions can be specified by splitting into a set of convex regions. For match regions, the contact must be in at least one region or else it is filtered out. For ignore regions, if the contact is any one (or more) regions, it will be filtered out. If the contact fails according to any kind filter, it fails overall. Here are some examples:

```
match_region  =  pts={10,-40:-50,-80:-30,-140:50,-100}
ignore_ignore =  pts={50,-60:80,-70:120,-110:100,-150:60,-160:30,-110}
```

If a contact survives the region filters at the contact manager level, it still must survive any filters that may have been declared at the individual alert level. The `match_region`, and `ignore_region` components work the same, but are embedded in an alert configuration. As a `pContactMgrV20` configuration parameter, it may look like:

```
alert = id=K8, on_flag=SAY_MOOS=alarm, off_flag=SAY_MOOS=ok,    \
        alert_range=20, cpa_range=30, match_group=blue_team     \
        match_region=pts={10,-40:-50,-80:-30,-140:50,-100}
```

As an alert configured by a mail request to the contact manager, it may look like:

```
BCM_ALERT_REQUEST = id=k8, onflag=SAY_MOOS=alert, alert_range=20, cpa_range=40, \
                    match_region=pts={10,-40:-50,-80:-30,-140:50,-100}
```

Note: Whereas type and group information tend not to change for a contact during operation, its position in or out of a region may very well change. If a contact that was previously being track, with an entry in memory, fails a region filter, it is also wiped from memory in the contact manager. This is to prevent contact information from becoming stale in memory.

### 5.3 Exclusion Filters Based on Contact Range to Ownship

Last but not least, the simplest exclusion filter is to ignore any contact substantially far from ownship. Using the `reject_range` parameter this range can be set from the configuration block of `pContactMgrV20`:

```
reject_range = 2000
```

The default value is 2000 meters. To disable this filter, either set it to a very large value, or give it the string value `"off"`. Keep in mind that the `reject_range` serves a second purpose, as discussed in Section 3.2, as a threshold for retiring a contact from the contact manager's internal data structures. Retiring occurs if either (a) the contact range exceeds the `reject_range`, or (b) if contact reports are stale beyond a certain time. By turning off the `reject_range`, contact retirement can only be caused by staleness. Presently there are no warnings produced if the `reject_range` is lower than any of the alert ranges. This check will likely be added in future releases.

## 6 Additional Contact Manager Reports

### 6.1 Closest Contact Reports

The contact manager will publish information about the contact currently with the closest range to ownship. It will publish four variables, `CONTACT_CLOSEST` naming the vehicle that is closest, `CONTACT_CLOSEST_TIME` indicating the time at which the named contact became the closest contact, `CONTACT_CLOSEST_RANGE` indicating the actual current range to the closest contact, and `CONTACT_RANGES` with an ordered list of ranges for all known contacts. For example:

```
CONTACT_CLOSEST = "vessel_0971"
CONTACT_CLOSEST_TIME = 7725262005.34
CONTACT_CLOSEST_RANGE = 48.32
CONTACT_RANGES = 48.32,56.3,112.08
```

The first two variables above in practice will not change very often, perhaps on the order of once per several seconds. The contact range however will likely change on every iteration. While negligible in terms of MOOS bandwidth, it may be annoying in terms of logged data, so it can be configured to be off, by setting the parameter `post_closest_range=false`. The same is true for the variable `CONTACT_RANGES`, which can be disabled by setting `post_all_ranges=false`. The default value for both is true. The rate of data logged can also always be adjusted in the `pLogger` configuration.

## 6.2 Customizable Contact Reports Based on a Range Threshold

The contact manager may optionally be configured to produce customizable contact reports based on a range threshold. These reports are simply a list of contacts. A subscriber to the contact manager could otherwise glean this information via the `CONTACTS_RECAP` message which contains the full list of all contacts and ranges. The customizable contact report described here is simply a more convenient low-bandwidth alternative option.

### 6.2.1 Format of the Contact Reports

The format of the report is simply a comma-separated list of contacts. For example:

```
CONTACTS_050 = vid_0027, vid_0197, vid_8645
```

This report is published only when the list of contacts changes, so it is considerably less bandwidth and easier to parse than the `CONTACTS_RECAP` message if the user simply wants to know which contacts are currently within 50 meters. Note the variable name, `CONTACTS_050`, was chosen with the `_050` suffix as an easy reminder of the meaning of the list. This is just a useful convention, and not enforced in any way. Selecting the name of the variable, and range is described next.

### 6.2.2 Requesting a Customized Contact Report

The MOOS variable used for posting the report, and the range threshold, are specified via the incoming MOOS variable `BCM_REPORT_REQUEST`. For example:

```
BCM_REPORT_REQUEST = "var=CONTACTS_050, range=50, group=blueteam"
BCM_REPORT_REQUEST = "var=CONTACTS_100, range=100"
BCM_REPORT_REQUEST = "var=CONTACTS_200, range=200, type=sailboat, refresh=true"
```

The order in the components in the string do not matter. It is recommended to use a MOOS variable naming convention, such as `CONTACTS_050` above, to indicate the meaning of the report. The variable name and range components are mandatory, while the group and type components are optional. If the group or type component is used, then a contact must satisfy all criteria to be included on the contact list for that report. When the `refresh=true` component is detected, a new report will be generated immediately, even if the content of the report has not changed from a previous posting.

### 6.2.3 Timeout of Custom Reports

While a valid custom report request will generate at least one custom report immediately, the generation of further reports will cease after some point in time unless further requests arrive. By default the duration is 10 seconds, but may be changed to say 60 seconds with the following parameter: `range_report_timeout=60`.

# 7 Guarding Against Unbounded Memory Growth

In some systems it may be possible for the sensors or sensor system to be generating many contacts, with unique IDs, over time. This problem may arise simply from long duration (days+) missions in high contact density situations. It may also be partly a result of a sensor that easily loses track on a contact, generating several IDs for the same contact as it approaches and ultimately departs the ownship vicinity.

The contact manager reasons about two types of data, tied to (a) active contacts, and (b) retired contacts. The key data structures were shown in Figure 3. In this section, the contact manager policies to guard against unbounded growth are described.

## 7.1 Managing Retired Contacts

As a contact is retired, it is moved to an internal list of retired contacts, and this list is published as `CONTACTS_RETIRED`. This is useful for debugging and verifying contact manager operation. With shore test missions having only a handful of contacts, this list can hold all contacts ever retired. In more realistic missions, the list needs to be managed.

As discussed in Section 4.3, the newly retired contact will stay on the `CONTACTS_RETIRED` list until it is pushed off, first-in first-out, when newer retired contacts are added. The size of the list is set by the `max_retired_history` parameter to be in the range $[1, 50]$, ensuring that even this list does not grow unbounded. The default is 5. The sequence of steps at the end of each iteration of the contact manager is the following:

- Add newly retired contacts to the retired list
- Publish the retired list to `CONTACTS_RETIRED`
- If needed, prune the retired list to size `max_retired_history`

This means that (a) the list published to `CONTACTS_RETIRED` on some iterations may be longer than `max_retired_history`, and (b) every contact that is ever retired will show up on at least one publication to `CONTACTS_RETIRED`.

## 7.2 Managing Active Contacts into Retirement

To have an absolute guard against unbounded memory, the contact manager has a strict upper limit on the number of contacts, configured with the parameter, `max_contacts`. This is set to 500 by default. Although most systems have enough memory to probably handle many thousands of contacts, 500 is probably generously high. The user takes responsibility to set this value.

### 7.2.1 Normal Retirement

As ownship and contacts pass each other, coming in and out range, or comms and sensor range, contact retirement proceeds simply. At the end of each contact manager iteration, after receiving all incoming contact updates and generating any applicable alerts or status postings, the contact manager will retire any contact that meets the following criteria:

- The contact is beyond the range set by the parameter reject_range, or
- The timestamp of the contact's most recent report is stale beyond the time set by the parameter max_contact_age.

The default value for reject_range is 2000 meters, and the default value for max_contact_age is 60 seconds. If, at this point, the number of contacts is still greater than max_contacts, then contacts will need to be forced into retirement, or *culled*.

### 7.2.2 Forced Retirement

If normal retirement doesn't achieve the desired result of reducing the total contacts to be less than or equal to max_contacts, the following two additional steps will be taken:

- A tighter maximum age is applied to each contact's staleness. Contacts with a timestamp larger than *half* the time set by the parameter of the max_contact_age will be removed.
- Finally, if the above step leaves more culling to be done, contacts will be removed based range, until only the N closest contacts remain, where N is equal to max_contacts.

### 7.2.3 The Consequences of Forced Retirement

A contact that has been forced into retirement may have been in the alerted state with respect to one or more alert types. The forced retirement means it did not have a normal transition from being alerted to unalerted. If there was an `off_flag` configured for the alert, it will not get posted as normal. This should be considered when/if using an `off_flag`.

If a culled contact was in an alerted state for one or more alert types, it might reappear a short time later as the contact manager again receives information about this contact. In fact, if the contact happens to be the (N+1)th contact in terms of range, it conceivably could reappear and be culled on successive iterations. What are the consequences of this in terms of alerts? The alert requests used by the contact-related behaviors are designed such that successive identical alerts are harmlessly redundant.

For example, consider the `on_flag` from the earlier example:

```
CONTACT_INFO=name=avd_vessel_023 # contact=vessel_02
```

If a contact behavior did not exist, one would be spawned as normal. If one had already been spawned, the helm would direct this update to that behavior (re)setting the `name` and `contact`

parameters to the same value as before. In either case the regeneration of the alert is either doing its job or is harmless.

A culled contact may however affect the accuracy of customized range reports. If the user has asked to be notified of all contacts within say 10,000 meters, but there are more than max_contact vessels within this range, then the range report may not be accurate.

# 8 Deferring to Earth Coordinates over Local Coordinates

Incoming node reports contain the position information of the contact and may be specified in either local x-y coordinates, or earth latitude longitude coordinates or both. By default pContactMgrV20 uses the local coordinates for calculations and the earth coordinates are merely redundant. It may instead be configured, with the contact_local_coords parameter, to have its local coordinates set from the earth coordinates if the local coordinates are missing:

```
contact_local_coords = lazy_lat_lon
```

It may also be configured to always use the earth coordinates, even if the local coordinates are set:

```
contact_local_coords = force_lat_lon
```

The default setting is `verbatim`, meaning no action is taken to convert coordinates. If either of the other two above settings are used, the latitude and longitude coordinates of the local datum, or `(0,0)` point must be specified in the MOOS mission file, with LatOrigin and LongOrigin configuration parameters. (They are typically present in all mission files anyway.)

# 9 Terminal and AppCast Output

The status of the contact manager may be monitored from from an open console window where pContactMgrV20 is launched. Example output is shown below in Listing 1.

*Listing 9.1: Example pContactMgrV20 terminal and appcast output.*

```
 1  ============================================================================
 2  pBasicContactMgr abe                                          0/0(377)
 3  ============================================================================
 4  X/Y from Lat/Lon:   false
 5  Contact Max Age:    3600
 6  Reject Range:       2000
 7  BCM_ALERT_REQUESTs: 2
 8  DisplayRadii:       false
 9
10  Alert Configurations (1):
11  ---------------------
12  1 ID: avdcol_
13    Alert Ranges: 35/45
14    Source: pHelmIvP
15    OnFlag: CONTACT_INFO=name=$[VNAME] # contact=$[VNAME]
```

```
16
17  Alert Status Summary:
18  ---------------------
19         List: ben,cal,deb
20      Alerted: (deb,avdcol_)
21      Retired:
22        Recap: vname=ben,range=82.72,age=0.83 # vname=cal,range=73.42,age=0.66 #
23               vname=deb,range=26.32,age=0.81
24
25  Contact Status Summary:
26  -----------------------
27  Contact     Range    Alerts    Alerts
28              Actual   Total     Active
29  -------     ------   ------    ------
30  ben           82.7   0         0
31  cal           73.4   1         0
32  deb           26.3   3         1
33
34  Custom Contact Reports:
35  -----------------------
36  VarName     Range     Group     VType     Contacts
37  -------     -----     -----     -----     --------
38
39  ==========================================================================
40  Most Recent Events (4):
41  ==========================================================================
42  [179.51]: CONTACT_INFO=name=deb # contact=deb
43  [72.09]: CONTACT_INFO=name=deb # contact=deb
44  [71.06]: CONTACT_INFO=name=deb # contact=deb
45  [33.44]: CONTACT_INFO=name=cal # contact=cal
```

On line 2, the `"0/0"` indicates there were no configuration warnings and no run-time warnings (thus far). The `"(377)"` represents the iteration counter of pContactMgrV20. Line 4 indicates that contact position in local coordinates is being derived directly from reported local coordinates in incoming node reports (Section 8). Line 5 shows the maximum age of a contact report before the contact is dropped (Section 3.2). Line 6 indicates the contact range beyond which it will be ignored (Section 5.3). Line 7 indicates the number of alert request messages that have been received (Section 5.2). Line 8 indicates whether visual circles are to be displayed showing the range and cpa range for a named alert (Section 12). In lines 10-15, the alerts configured by the mission file or incoming requests are shown. If multiple alerts types are configured, they would each be listed here separated by their alert id.

In lines 17-23, the alert status of the contact manager is output. These five lines are equivalent to the content of the CONTACTS_* variables described in Section 3.3. The block beginning line 34 would show any custom contact report configurations (Section 6.2), but there are no custom reports in this example. In lines 25-32, the status and alert history for each known contact is shown. Finally, starting on line 42, a limited list of recent events is shown. In this example, four alerts are shown, three for the vehicle `deb`, and one for the vehicle `cal`. In this particular mission, vehicles routinely proceed in and out of range to ownship, so the helm has been alerted about `deb` three times.

# 10 Configuration Parameters for pContactMgrV20

The following parameters are defined for pContactMgrV20. A more detailed description is provided in other parts of this section. Parameters having default values are indicated so.

*Listing 10.2: Configuration Parameters for pContactMgrV20.*

| | |
|---|---|
| alert: | A description of a single alert. Section 2.2. |
| alert_verbose: | If true, verbose debugging output will be published to the variable ALERT_VERBOSE. The default is `false`. |
| contact_max_age: | Seconds between reports before a contact is retired and dropped from the list. Legal values: any non-negative number. The default is 600. Sections 4.3, and 7. |
| contact_local_coords: | Determines if the local coordinates of incoming node reports are filled by translated latitude longitude coordinates. Legal values: `verbatim`, `lazy_lat_lon`, `force_lat_lon`. The default is `verbatim`, meaning no translation action is taken. Section 8. |
| decay: | An optional way of treating delays in between incoming contact reports. Legal values are comma-separated pair of numbers, e.g., `"15,30"`, where the first number represents the interval of time where the contact position will be extrapolated linearly into the future given its last known heading and speed. After 15 seconds, the contact speed used for extrapolation will linearly reduce to zero at 30 seconds. The default value is `"15,30"`. Extrapolation may be turned off by setting this value to `"0,0"`. |
| alert_range_color: | The default color for rendering the alert range radius. Legal values: any color in Colors Appendix . The default is *gray70*. Section 12. |
| cpa_range_color: | The default color for rendering the cpa range radius. Legal values: any color in the Colors Appendix. The default is *gray30*. Section 12. |
| display_radii: | If true, the two alert ranges are posted as viewable circles. Legal values: `true`, `false`. The default is `false`. |
| display_radii_id: | If multiple alert IDs are configured, this parameter can be used to indicate which alert will have its range circles rendered when rendering is enabled. Section 12. |
| ignore_group: | A comma-separated list of contact groups to ignore, defined at the global contact manager level. Section 5.1. |
| ignore_name: | A comma-separated list of contact names to ignore, defined at the global contact manager level. Section 5.1. |
| ignore_region: | A convex polygon wherein contacts will be ignored, defined at the global contact manager level. Section 5.2. |
| ignore_type: | A comma-separated list of contact types to ignore, defined at the global contact manager level. Section 5.1. |
| match_group: | A comma-separated list of contact groups to match, defined at the global contact manager level. Section 5.1. |

match_name: A comma-separated list of contact names to match, defined at the global contact manager level. Section 5.1.

match_region: A convex polygon outside of which contacts will be ignored, defined at the global contact manager level. Section 5.2.

match_type: A comma-separated list of contact types to match, defined at the global contact manager level. Section 5.1.

max_retired_hist: Maximum amount of contacts held in the contact managers memory at any given time before increasingly more dire measures are taken to remove/retire contacts. Section 7.

max_retired_hist: Limit the number of contacts listed in the CONTACTS_RETIRED list to this number. The default is 5, and legal values are in the range $[1, 50]$. Regardless of this setting, all retired contacts will be included in a CONTACTS_RETIRED posting at least once. (Section 3.2).

post_all_ranges: If true, the range to all contacts is posted in an ordered, comma-separated list. The default is `false`. Section 6.1.

post_closest_range: If true, the range to the closest contact is posted, rounded to the nearest meter. The default is `false`. Section 6.1.

recap_interval: Sets a threshold number of seconds between successive postings of the CONTACTS_RECAP variable. This variable may be just too verbose for certain use cases, and if logged, can contribute to log file bloat. The default is one second. It can be shut off completely by setting it to string value `"off"`. Section 3.3.

reject_range: A range, in meters, beyond which new contacts will be ignored, and existing contacts will be dropped. This is one of the guards against unbounded memory growth, discussed in Section 7. The default value is 2000 meters.

strict_ignore: if true, both group and type exclusion filtering will reject a contact if the contact is not reporting its group or type information and there are ignore filters. Section 5.1.

## 10.1 An Example MOOS Configuration Block

To see an example MOOS configuration block, enter the following from the command-line:

```
$ pContactMgrV20 --example or -e
```

This will show the output shown in Listing 3 below.

*Listing 10.3: Example configuration of the pContactMgrV20 application.*

```
1  ============================================================
2  pContactMgrV20 Example MOOS Configuration
3  ============================================================
4
5  ProcessConfig = pContactMgrV20
```

```
 6  {
 7    AppTick   = 4
 8    CommsTick = 4
 9
10    // Alert configurations (one or more, keyed by id)
12    alert = id=avd, onflag=CONTACT_INFO="name=avd_$[VNAME] # contact=$[VNAME]"
13    alert = id=avd, range=80, alert_range_color=white
14    alert = id=avd, cpa_range=95, cpa_range_color=gray50
14    alert = id=avd, ignore_name=hal, match_group=usv,ship
15
22    // Policy for retaining potential stale contacts
23    contact_max_age  = 600             //  default in secs.
24    max_retired_history = 5            //  default in # of ships
25
26    // Configuring other output
27    display_radii     = false  // or {true}
28    alert_verbose     = false  // If true, ALERT_VERBOSE published.
19    alert_range_color = color  // default is gray65
20    cpa_range_color   = color  // default is gray35
29
30    // Policy for linear extrapolation of stale contacts
31    decay = 15,30                       // the default in secs
32
32    reject_range   = 2000      // default is 2000 (meter)
33    recap_interval = 1         // default in 1 (second)
34
35    contact_local_coords    = verbatim  // default is verbatim
38    post_closest_range      = false     // default is false
39  }
```

# 11 Publications and Subscriptions for pContactMgrV20

The interface for pContactMgrV20, in terms of publications and subscriptions, is described below. This same information may also be obtained from the terminal with:

```
$ pContactMgrV20 --interface or -i
```

## 11.1 Variables Published by pContactMgrV20

The primary output of pContactMgrV20 to the MOOSDB is the set of user-configured alerts. Other variables are published on each iteration where a change is detected on its value:

- APPCAST: Contains an appcast report identical to the terminal output. Appcasts are posted only after an appcast request is received from an appcast viewing utility. Section 9.
- ALERT_VERBOSE: Verbose or debugging output published to this variable if alert_verbose is set to true.
- CONTACT_CLOSEST: Name of the closest contact. Section 6.1.
- CONTACT_CLOSEST_TIME: The time at which the closest contact became the closest contact. Section 6.1.
- CONTACT_CLOSEST_RANGE: The range of the closest contact. Section 6.1.

- CONTACT_MGR_WARNING: A warning message indicating possible mishandling of or missing data.
- CONTACT_RANGES: A comma-separated and ordered list of all contact ranges. Section 6.1.
- CONTACTS_ALERTED: A list of contacts for which alerts have been posted. Section 3.3.
- CONTACTS_COUNT: The number of contacts currently alerted. Section 3.3.
- CONTACTS_LIST: A comma-separated list of contacts. Section 3.3.
- CONTACTS_RECAP: A comma-separated list of contact summaries. Section 3.3.
- CONTACTS_RETIRED: A list of contacts removed due to the information staleness. Section 3.3.
- CONTACTS_UNALERTED: A list of contacts for which alerts are pending, based on the range criteria. Section 3.3.
- PCONTACTMGR_PID: The system process ID of the contact manager.
- VIEW_CIRCLE: A rendering of the alert ranges. Section 12.

Some examples:

```
CONTACTS_LIST      = gus,joe,ken,kay
CONTACTS_ALERTED   = gus,kay
CONTACTS_UNALERTED = ken,joe
CONTACTS_RETIRED   = bravo,foxtrot,kilroy
CONTACTS_RECAP     = name=gus,age=7.3,range=13.1 # name=ken,age=0.7,range=48.1 # \
                     name=joe,age=1.9,range=73.1 # name=kay,age=4.0,range=18.2
```

## 11.2 Variables Subscribed for by pContactMgrV20

The pContactMgrV20 application will subscribe for the following MOOS variables:

- APPCAST_REQ: A request to generate and post a new appcast report, with reporting criteria, and expiration.
- BCM_ALERT_REQUEST: If false, no postings will be made for rendering the alert and cpa range circles. Section 2.3.
- BCM_DISPLAY_RADII: If false, no postings will be made for rendering the alert and cpa range circles. Section 12.
- BCM_REPORT_REQUEST: If false, no postings will be made for rendering the alert and cpa range circles. Section 6.2.2
- NAV_HEADING: Present ownship heading in degrees.
- NAV_SPEED: Present ownship speed in meters per second.
- NAV_X: Present position of ownship in local $x$ coordinates.
- NAV_Y: Present position of ownship in local $y$ coordinates.
- NODE_REPORT: A report about a known contact.

## 11.3 Command Line Usage of pContactMgrV20

The pContactMgrV20 application is typically launched as a part of a batch of processes by pAntler, but may also be launched from the command line by the user. To see command-line options enter the following from the command-line:

```
$ pContactMgrV20 --help or -h
```

This will show the output shown in Listing 4 below.

*Listing 11.4: Command line usage for the pContactMgrV20 application.*

```
1  ============================================================
2  Usage: pContactMgrV20 file.moos [OPTIONS]
3  ============================================================
4
5  SYNOPSIS:
6  ------------------------------------
7    The contact manager deals with other known vehicles in its
8    vicinity. It handles incoming reports perhaps received via a
9    sensor application or over a communications link. Minimally
10   it posts summary reports to the MOOSDB, but may also be
11   configured to post alerts with user-configured content about
12   one or more of the contacts.
13
14 Options:
15   --alias=<ProcessName>
16       Launch pContactMgrV20 with the given process
17       name rather than pContactMgrV20.
18   --example, -e
19       Display example MOOS configuration block.
20   --help, -h
21       Display this help message.
22   --interface, -i
23       Display MOOS publications and subscriptions.
24   --version,-v
25       Display the release version of pContactMgrV20.
26
27 Note: If argv[2] does not otherwise match a known option,
28       then it will be interpreted as a run alias. This is
29       to support pAntler launching conventions.
```

# 12 Visuals

The contact manager will optionally generate visual artifacts, in the form of postings to the variable VIEW_CIRCLE, for consumption by pMarineViewer for rendering the alert range and cpa range. Rendering of ranges is done with the following three configuration parameters:

```
diplay_radii = true
alert_range_color = <color>
cpa_range_color = <color>
```

The default value for display_radii is false. The default value for alert_range_color is gray70, and the default value for cpa_range_color is gray30. See the Colors Appendix for more on colors.

If multiple alerts are configured in the contact manager, the user may need to specify which alert ID to render. If no alert ID is specified, the contact manager will choose the first one in its memory. To specifify the ID, the following parameter may be used:

```
diplay_radii_id = <alert_id>
```

The posting of rendering circles by pContactMgrV20 may also be be adjusted dynamically via the incoming MOOS message BCM_DISPLAY_RADII. The value of this message may be either

- `true` or `on`,
- `false` of `off`,
- `toggle`,
- A configured alert ID