MÄK VR-Forces 3.7.1 Release Notes

This release note contains the following release-specific information for VR-Forces Release 3.7.1:

Document ID: VRF-3.7.1-3-040909

Systems Supported

VR-Forces 3.7.1 is available for the platforms listed in Table 1. For the availability of other platforms, contact your MÄK salesperson. For toolkit users, application code must be built with the indicated compilers in order to link to VR-Forces libraries.

Table 1: Platforms supported

Operating System	Compiler
SGI IRIX6.5	MipsPro7.4.2 C++ compiler (available from SGI)
Red Hat Linux 9.0 Red Hat Enterprise Linux WS release 3 for Intel x86	GNU C++ (GCC) 3.2.2
Windows 2000 SP2, XP	Microsoft Visual C++ 6.0, Service Pack 5 Microsoft .NET 7.1

If you are building VR-Forces with .NET on Windows 2000, you must install the latest version of the directX SDK (obtainable from Microsoft).

Patching the Microsoft Visual C++ 6.0 Compiler

VR-Forces is transitioning to use of the Standard Template Library, particularly for lists and map objects. Microsoft Visual C++ version 6.0 has a bug with the STL that is fixed with the supplied *xtree-msvc++patch* file. If you use MS Visual C++ version 6.0, you must install a patch to a standard header file. The patch comes from Dinkumware, Ltd (the company that developed the Standard C++ Library header files for Microsoft), but for convenience, we are distributing it with VR-Forces releases.

The patch fixes a bug that would cause applications to crash if they pass std::sets or std::maps from an application to a DLL or vice-versa, something that IEEE 1516 federates must do. For details about the patch go to http://www.dinkumware.com/vc_fixes.html.

To install the patch:

1. In the top level VR-Forces directory, is a file called *XTREE-msvc++patch*. Rename this file *XTREE*.
2. Make a back-up version of the original *XTREE*.
3. In the Visual Studio *include* directory (for example, *C:\Program Files\Microsoft Visual Studio\VC98\Include*), replace the original version of *XTREE* with the patched version.

Building on Linux or IRIX

The example Makefiles require a patched version of gmake 3.80, which has been included for your convenience in the *./mkbin* directory. This version of gmake is based on version 3.80, but includes a memory fix that is needed to use our makefiles. The source code for the modified gmake is freely available from *http://ftp.mak.com/out/gmake3.80-patch1.tar.gz*. This patch will be included in future versions of gmake.

Before you compile the examples, go to the top level of your installation and create a symbolic link 'VR-Link' to your VR-Link installation and another called 'RTI' to your RTI installation.

Running VR-Forces on SGI IRIX

If you want to run VR-Forces on SGI IRIX, you must set the color depth to 24-bit.

True Type fonts are not supported by SGI IRIX. Use bitmaps for entity icons.

Disk Space Requirements

A full installation of VR-Forces requires approximately 540 MB of disk space. Terrain databases use approximately 106 MB and documentation (primarily the class reference) uses 200 MB.

MÄK Product Compatibility

To build VR-Forces applications, you must link with VR-Link 3.9.1. If you are building for HLA and want to link with the MÄK RTI, use MÄK RTI 2.x.

Qt Release Compatibility

If you want to do development using the GUI API, you need to use Qt, a cross-platform GUI toolkit. The VR-Forces GUI was built with Qt release 3.3.1. A VR-Forces developer's license includes a Qt development license. MÄK provides you with a Qt license key. However, you must download the correct version of Qt from *www.trolltech.com*.

Third-Party Library Requirements

DtTerrainProfileWidget is based, in part, on the work of the Qwt project (*http://qwt.sf.net*). To compile examples or user projects, you must download the QWT distribution listed on their home page. Set the MAK_QWTDIR environment variable to point to the location of the QWT library.

Network Compatibility

HLA only

VR-Forces 3.7.1 was built against VR-Link 3.9.1 and is compliant with:

- RPR-FOM 1.0 and a subset of 2.0 (draft 6 and 17)
- MÄK RTI 2.2
- DMSO RTI NG 6.

If you need to run the Linux version of VR-Forces with the DMSO RTI, contact MÄK technical support.

VR-Forces 3.7.1 is compatible with applications that use earlier versions of VR-Link if they support versions of the RPR FOM listed above.

DIS only

VR-Forces 3.7.1 supports DIS 2.0.4, IEEE 1278.1, 2.1.4, and IEEE 1278.1a, and can therefore interoperate with DIS applications of any of these versions.

Backwards Compatibility

VR-Forces 3.7.1 is compatible with VR-Forces 3.7 and MÄK Plan View Display 2.7.

RPR FOM Versions Supported

VR-Forces 3.7.1 has built-in support for versions 1.0 and 2.0 (draft 6 and 17) of the RPR FOM. It also supports VR-Link's ability to support alternative FOMs through the FOM Mapper. By default, VR-Forces 3.7.1 uses RPR FOM 1.0.

If you are building an application with the VR-Forces toolkit and you want to specify a version of the RPR FOM through code, pass the version number (for example, 2.0006) to the *DtRprFomMapper* constructor and pass the resulting object to the *DtExercise-Conn* constructor. Also, make sure you are using a federation execution name that corresponds to the right FED file. For example:

```
DtExerciseConn conn("VR-Link20006", "MyApp", new DtRprFomMapper(2.0006));
```

In order to support RPR FOM 1.0, we have added the following extensions (which are supported in later versions of the RPR FOM):

- EnvironmentProcess
- GriddedData
- EmbeddedSystem.IFF
- EmbeddedSystem.IFF.NatoIFF
- EmbeddedSystem.IFF.NatoIFF.NatoIFFTransponder
- EmbeddedSystem.IFF.NatoIFF.NatoIFFInterrogator
- EmbeddedSystem.IFF.SovietIFF

- EmbeddedSystem.IFF.SovietIFF.SovietIFFTransponder
- EmbeddedSystem.IFF.SovietIFF.SovietIFFInterrogator
- EmbeddedSystem.IFF.RRB
- BaseEntity.AggregateEntity
- ObjectRoot.BaseEntity.PhysicalEntity.Lifeform.

For both RPR FOM 1.0 and 2.0 VR-Forces 3.7.1 relies on the LgrControl and View-Control custom MÄK extensions.

New Features and Updates

This release of MÄK VR-Forces is primarily a maintenance release. It also has the following updates and new features:

- New examples
- New configuration parameters
- Cruise missile model
- Copying parameter entries
- API changes
- Updating object parameters databases.

New Examples

VR-Forces has the following new examples:

- envProcessListen – demonstrates how to listen for and display state updates for environmental process objects (graphical objects) generated by VR-Forces.
- addPSR – demonstrates how to add a process state repository to a component to allow its internal state to be checkpointed to the order of battle file.

New Configuration Parameters

VR-Forces has new configuration parameters to control how VR-Forces reads messages from the network. The parameters affect the DtExerciseConn::drainInput() member function. This function causes VR-Link to read and process input from the network.

- **VRF_maxDrainInputTime** – sets the maximum time to wait in drainInput() before returning. For the DMSO and MÄK RTIs, the default values will drain the entire input stream before returning. For the Pitch RTI, the default values read a single input from the input stream, so set to a non -1 value to process more information. Use with DIS or HLA. Use only with the VR-Forces back-end.

  ```
  (setqb VRF_maxDrainInputTime -1.0)
  ```

- **VRF_minDrainInputTime** – The minimum time to stay in drainInput() (HLA only). Use only with the VR-Forces back-end.

  ```
  (setqb VRF_minDrainInputTime 0.0)
  ```

- **VRF_maxDrainInputReads** – The maximum number of reads to make in drainInput() before returning (DIS only). Use only with the VR-Forces back-end.

  ```
  (setqb VRF_maxDrainInputReads 100)
  ```

- **PVD_maxDrainInputTime** – sets the maximum time to wait in drainInput() before returning. For the DMSO and MÄK RTIs, the default values will drain the entire input stream before returning. For the Pitch RTI, the default values read a single input from the input stream, so set to a non -1 value to process more information. Use with DIS or HLA. Use only with the VR-Forces front-end.

  ```
  (setqb PVD_maxDrainInputTime -1.0)
  ```

- **PVD_minDrainInputTime** – The minimum time to stay in drainInput() (HLA only). Use only with the VR-Forces front-end.

  ```
  (setqb PVD_minDrainInputTime 0.0)
  ```

- **PVD_maxDrainInputReads** – The maximum number of reads to make in drainInput() before returning (DIS only). Use only with the VR-Forces front-end.

  ```
  (setqb PVD_maxDrainInputReads 100)
  ```

Support for Cruise Missiles

VR-Forces now supports cruise missiles. A cruise missile can fly directly to a target or follow a route.

To fire a cruise missile:

1. Select an entity that has cruise missile resources.
2. Choose Task → Fire Cruise Missile. The Fire Cruise Missile dialog box opens.

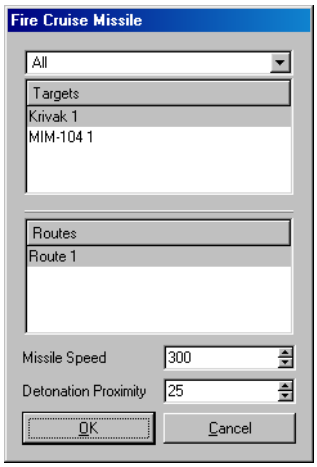

Figure 1. Fire Cruise Missile dialog box

3. Select the target from the list or on the map. Targets can be entities or points.
4. Optionally, select a route for the missile to follow. If no route is selected, the missile flies directly to the target.

5. Specify the ordered speed for the missile.
6. Specify the detonation proximity for the warhead. It will detonate within this range of the selected target.
7. Click OK.

The Fire Cruise Missile task is listed on the Task menu for all entities. However, it is only meaningful when assigned to entities with cruise missile launching capability.

Changes to the VR-Forces API to Support Cruise Missiles

Cruise missiles are basically a special type of fixed-wing entity. To support them, the fixed-wing components have been modified and new components added.

New Parameters

The Fixed Wing Pilot Controller and all derived controllers, such as move-to, move-along, and follow-entity, have a new parameter called **max-pitch-angle**. It specifies the maximum pitch angle (in radians) which the pilot will use when changing altitudes in this aircraft. This has no relation to actual capabilities of the aircraft model, but only how it will be flown. Default: 0.75.

The Damage Actuator has a new parameter called **terminate-on-destroy**. If set to true, the entity is removed from the simulation as soon as it receives a catastrophic kill. Default: False.

Warhead Detonation Actuator

The warhead detonation actuator, *DtWarheadDetonationActuator*, models the detonation of a warhead that is mounted on an entity, for example, a warhead on a missile. The warhead can detonate based on entity, ground, or water impact, and/or based on proximity to a specific target entity or object. Upon detonation, a detonation interaction is created using parameters passed in using the Set Warhead Info set data request. The host entity is also destroyed and removed from the simulation.

The following parameters are set in the *DtWarheadDetonationActuatorDescriptor*:

- **detonate-on-ground-impact** – If true, a terrain impact with non-water terrain triggers the detonation. Default: True.
- **detonate-on-water-impact** – If true, a terrain impact with water terrain triggers the detonation. Default: True.
- **detonate-on-entity-impact** – If true, any time the host entity's bounding volume intersects with any other entity's bounding volume, the detonation is triggered. Default: True.

The following parameters are configured for an individual detonation actuator by sending the *DtSetWarheadInfo* set data message to the host entity:

- Warhead Type – The enumeration of the warhead type to use in the detonation interaction.
- Fuze Type – The enumeration of the fuze type to use in the detonation interaction.
- Attacker Name – The object name of the attacker to use in the detonation interaction. If specified, the warhead does not detect collisions with the attacking entity. (Presumed to be the launching entity, if this is a munition.)
- Event Id – The string representation of the event ID to use in the detonation.
- Target Name – The name of the target entity, if proximity detonate is desired.
- Detonation Proximity – The proximity to the target that triggers a detonation.

The Warhead Detonation Actuator does not have any ports.

Cruise Missile Launch Controller

The cruise missile launch controller, *DtCruiseMissileLaunchController*, models a launcher of missiles fired only on external orders. The controller responds to *DtFireCruiseMissile* tasks, and creates a new entity of the type specified in its descriptor.

When a *DtFireCruiseMissile* task is received, the controller calls startTask() to store the task, and returns. The task is processed by the tick() function.

When the controller is ticked:

1. If there is a current task, the simulation is not paused, and the minimum time between launches has elapsed, it continues. Otherwise it returns.
2. It calls missileResourceAvailable() to make sure it has not run out of missiles. Returns if this method return false.
3. It calls consumeMissileResource() to decrement the missile resource.
4. If calls createMissileObject(), which returns a pointer to a newly created simulation object which is the missile being launched.
5. It creates and sends a *DtSetWarheadInfo* message with the host entity specified as the attacker, the target and proximity information from the *DtFireCruiseMissile* task, and the event ID, which is saved for use in the fire interaction created later in the tick function.
6. It calls createMissilePlan(), which returns a DtIfPlan for the missile. The plan message is then sent to the missile entity. The createMissilePlan() function uses the target, route, and speed information in the Fire Cruise Missile task to generate a plan to set the ordered speed, move along the route, then crash into the target.
7. It calls createAndSendFireInteraction() to create a fire interaction which is sent to the network.

The following parameters are set in the *DtCruiseMissileLaunchControllerDescriptor*:

- **munition-type** – The entity type of the missile to be created.
- **munition-name** – The name of the resource to decrement when a missile is launched.
- **launch-offset** – The position (relative to the host entity) where the missile will initially be placed.
- **launch-orientation** – The initial orientation (Tait-Bryan, relative to the host entity) for the missile.
- **min-time-between-launch** – The minimum simulation time between missile launches.

The Warhead Detonation Actuator does not have any ports.

Fixed-Wing Fly-Into Controller

The Fixed-Wing Fly-Into Controller, *DtFixedWingFlyIntoController*, is a derived type of fixed-wing move-to controller that tries to steer a fixed-wing entity directly into a target entity or object. It responds to *DtCrashInto* tasks. It is designed to hit ground entities from above. This controller contrasts with the typical fixed-wing behavior, in which a fixed-wing entity given a Move To task immediately flies to the altitude of the target point, then moves at that altitude towards the target.

When a *DtCrashIntoTask* is received, the ordered altitude of the fixed wing entity is set to either the current altitude of the entity, or the altitude of the target, whichever is greater.

When this controller is ticked and has an active task it:

1. Checks to see if the target position is within the **direct-fly-range**, or within the **direct-fly-elevation-angle**. Whenever the elevation angle to the target is greater than the **direct-fly-elevation-angle**, it is said to be within the angle.
2. If either of these are the case, the controller calls flyToDirect() in the parent *DtFixedWingPilotController* class. This causes the fixed wing entity to orient itself to point directly toward the target location and fly at the ordered speed.
3. If the target object is not in range, the controller calls flyTo(), which causes the fixed wing entity to move in the direction of the target while maintaining the ordered altitude.

This controller will never report task complete.

The following parameters are set in the *DtFixedWingCrashIntoControllerDescriptor* (in addition to all *DtFixedWingMoveToController* parameters):

- **direct-fly-elevation-angle** – The minimum elevation angle from the host entity to the target object which will trigger the direct-flight behavior of this entity.
- **direct-fly-range** – The minimum range between the host entity and the target object which will trigger the direct-flight behavior of this entity.

The Fixed-Wing Fly-Into Controller has a fixed-wing-control output port.

Changes to Existing Components

The Damage Actuator (*DtDamageActuator*) now checks for firepower-kill and mobility-kill entries in the damage files. These probabilities are specified exactly as the catastrophic-kill entry. The mobility-kill and firepower-kill attributes of the entity are set accordingly.

The Fixed Wing Actuator (*DtFixedWingActuatorComponent*) now calls fallFromSky() when the fixed wing entity runs out of fuel or has its mobility-kill appearance attribute set to true.

The Fixed Wing Pilot Controller (*DtFixedWingPilotController*) API has changed slightly with the addition of the flyToDirect() member function. You can call flyToDirect() instead of flyTo(), and it results in different behavior. The flyTo() function retains its original behavior of flying toward a target position while maintaining the ordered altitude. The flyToDirect() function orients the aircraft to point directly toward the destination location, and fly toward it.

New Tasks and Set Data Requests

VR-Forces has two new tasks and a set data request to support cruise missiles. *DtSetWarheadInfo* (*setWarheadInfo.h*) and *DtCrashIntoTask* (*crashIntoTask.h*) are used programmatically to plan the behavior of a cruise missile after a Fire Cruise Missile task is issued. They are not visible to a user through the GUI. *DtFireCruiseMissileTask* (*fireCruiseMissileTask.h*) is implemented as the Fire Cruise Missile task in the Task menu of the VR-Forces GUI.

Copying Parameter Entries

The information in this section was included in VR-Forces 3.7 Release Notes, but did not receive sufficient emphasis. It changes the behavior described in the last sentence of the first paragraph of section 3.3.2 of *MÄK VR-Forces Developer's Guide*.

For performance and memory usage reasons, the copy of the parameter entry that used to be made when creating a state repository is no longer made when the state repository is initialized. The **myParameterEntry** member variable points to the original parameter entry in the object parameters database and should only be accessed by the `const DtVrfObjectParameters* parameterEntry() const` member function.

If you want to get a copy of the parameter entry for modification, do not access the **myParameters** entry directly, because it will be NULL on access until a copy is created. This copy is created by calling the non const parameters() member function to return the **myParameters** entry. If a copy does not exist, one will be created by calling the createParameters() member function (at which point you can set the **myParameters** entry to your new parameter instance – this should be the only time you access the **myParameters** member variable directly). The parameters will then be initialized and setupParameterInheritance() will be called.

If you try to access the **myParameters** variable directly, the program might crash at runtime because it might be NULL, so your code should be written (or modified) to only call the member function that will return the information you want:

`const DtVrfObjectParameters* parameterEntry() const` returns a const pointer to the original parameter entry
`const DtVrfObjectParameters* parameters() const` also returns a const pointer to the original parameter entry. This member function is provided for backwards compatibility only and could be deprecated in future releases.
`DtVrfObjectParamters* parameters()` returns a mutable copy of the original parameter entry. Changes to this entry will not effect the original pointer.

API Changes

In the *DtFixedWingActuatorComponent* class, the data member **myflapsDeployedPort** was changed to **myFlapsDeployedPort**.

An accessor for the *DtExerciseConn* was added to *DtVrfApp*.

The **clamped-frame-rate** parameter has been added to the rotary wing descriptor object (component descriptor type **rotary-wing-component-descriptor**). This parameter is the frame rate threshold in Hertz at which the *DtRotaryWingActuator* will be clamped. When the frame rate drops below this threshold in the actuator, the actuator artificially clamps the frame rate to this value, enabling the model to still function reasonably at low frame rates. The default value for this parameter is 5.0 Hertz. A value of 0.0 Hertz indicates no clamping should be applied.

Updating Object Parameters Databases

The *VR-Forces 3.7 Release Notes* describe how to update object parameters databases to the new format. This section provides additional details for using the *convertParamDb* utilty.

The *convertParameterDb* executable is in the *./bin5*, *./binHLA13*, and *./binHLA1516* directories and not in the *./bin* directory, as documented in the *VR-Forces 3.7 Release Notes*. All three executables work the same way, and you can use any one of them to convert a given parameter database.

The *ammoselect*, *damage*, *detection*, *formation*, and *hit* directories referenced in the old object parameters database file must be available in their original paths (for example, *vrf3.6/data/parameters/ammoselect*) during conversion so that they can be copied to the new subdirectory created for the OPE files. Therefore, before running *convertParameterDb*, copy your old-format object parameters database file and the *ammoselect*, *damage*, *detection*, *formation*, and *hit* folders to the *./data/parameters* directory.

The *convertParamDb* utility must be run from the directory that contains the executable. For example, with the current working directory set to *vrf3.7/bin5*, a possible command line would be:

```
convertParamDb ../data/parameters/old.opd
  ../data/parameters/new.opd
```

Documentation Updates

This section lists updates to VR-Forces documentation, as indicated:

- The alert box on page 6-6 of *MÄK VR-Forces Developer's Guide* should read as follows:

!

Make sure that your net structure is padded out to an 8 byte boundary (the total size of the structure in bytes must be a multiple of 8). This prevents misalignment when the structure is included in another network structure.

Each element in your net structure must also be aligned on the byte boundary corresponding to its size. For instance a DtNetU16 must be on a 16 byte boundary. If it is preceeded in the structure by an DtNetU8, you must add an additional DtNetU8 of padding to align the DtNetU16 on a 16 byte boundary.

- The code sample on page 11-11 of *MÄK VR-Forces Developer's Guide* creates a DtVrfGuiFactory. It should create a *DtVrfGuiDefaultFactory*.
- You can use the Set Altitude request and can specify the Z value in a Set Location request for ground entities. Setting a ground entity's altitude allows you to place it at the various levels (floors) in building features.
- To clarify section 4.4.1 in *MÄK VR-Forces User's Guide*, note that the `-L` and `-r` startup options are for use only with the back-end, running in separate mode. In general, startup options specified as being back-end only, require separate mode, because in combined mode, you are really running *vrfGui* as the primary application.
- When you load a symbol map file through the GUI, only the symbols mapped in the symbol map file get replaced. If the symbol map file represents a subset of the total number of entity types being displayed, only that subset uses new symbols. The others will continue to use the previous symbols. This means that you may have a mixture of symbol types displayed. Furthermore, the Load Symbol Map option only accepts configuration files named *symbolmap-<xxx>.mtl*.

Bug Fixes

This release fixes the following problems:

- If a route is deleted while an entity is moving along it, the entity no longer continues to move along the former path of the route.
- Fixed a problem in which the event ID of detonation interactions sent out when a missile collided with the terrain were assigned a new ID instead of the event ID of the associated fire interaction.
- Shape files containing NULL records are now imported properly.
- DtVrfProcessStateRepositoryManager::removeProcessStateRepository no longer can go into an endless loop.
- An incomplete #ifdef in *vrfObjBaseSR.h* has been fixed.
- Fixed packet flood caused by the turret actuator when the host entity was destroyed.
- Entities no longer consider themselves to be under fire as a result of their own detonation interactions.
- Disconnecting and reconnecting a ballistic gun controller's trigger port no longer causes an accidental firing of the weapon.
- Enabled creation of the **myFlapsDeployedPort** in the *DtFixedWingPilotController* and the *DtFixedWingActuatorComponent* classes. Neither class currently makes use of this port in the fixed-wing model. It is provided so that toolkit users can take advantage of it.
- Rotary wing entities are now able to ascend vertically from a landed state. Previously, when tasked to move straight up (that is, given a move to altitude task), they would immediately land again.
- Dismounted infantry no longer fire through walls.

Known Problems

VR-Forces Release 3.7.1 has the following known problems:

- If you are using VR-Forces in a DIS exercise, you may experience problems with dropped packets and entities may time out. To avoid this problem, you can use asynchronous I/O by starting with the `-I` parameter.
- Aggregates composed of fixed-wing entities do not carry out tasks as an aggregate.
- When you aggregate a set of aggregates into a higher-echelon aggregate using the Aggregate As menu item, make sure that the aggregates being combined have been collapsed to their highest level. Failure to do so will aggregate the entities at the level they are currently displayed. For example, suppose that you want to aggregate two teams of two tanks each into a platoon. If at the time of the Aggregate As operation, one of the teams is expanded, displaying its constituent tanks, while the other team is collapsed, displaying a single team icon, the resulting platoon will be composed of two individual tanks and a team, instead of two teams.

- Fixed-wing take-off, landing, and taxiing behaviors may be difficult to configure through the front-end when simulating on geocentric terrain databases. This is due to the fact that it uses a different projected terrain database in the front-end when simulating on a geocentric terrain database in the back-end. Correlation errors in terrain elevation between the two databases must be small (on the order of meters) when placing waypoints or routes on the ground to support fixed-wing ground behaviors. Either use high resolution terrain databases (minimizing the elevation error between the front-end and back-end databases) when configuring these behaviors, or restrict your scenarios to fixed-wing flying behaviors.
- Using the *tdbtool* 's balancing option (`-b br,lf`) to balance certain terrains may cause the generated *.gdb* file to display incorrectly. This has to do with a draw ordering bug that is not currently corrected for all terrains. The resulting GDB file will still give correct intersection information, so it will still be suitable for simulation purposes.
- Surface entities are not configured to fire weapons. You can customize the object parameters database entries to enable this.
- The Qt translation files (in *./translations*) for built-in Qt-level GUI items do not work properly. The translation files for VR-Forces work correctly.
- The Set Formation request does not work properly on aggregate entities that are subordinate members of larger aggregate units.
- If you are running VR-Forces on Linux with the DMSO RTI, you cannot run in combined mode. Start the GUI and back-end separately.
- The order in which you create plans for aggregate units and individual members of aggregate units affects how the plans are implemented. If you create a plan for an aggregate member and then create a plan for the aggregate, the aggregate plan overrides the individual plan. If you create a plan for an aggregate and then create a plan for a member of the aggregate, the member plan is executed.
- Aggregates that have members simulated on multiple back-ends do not always act correctly. Remote subordinates do not break formation when they are independently tasked and do not report tasks for assumption if they get destroyed.
- If you run a MÄK Logger recording of a VR-Forces simulation and the site:application number for the back-end is the same as the site:application number of the back-end used to create the recording, VR-Forces may crash. To avoid this problem make sure that you run VR-Forces with different site:application numbers, or just run the front-end and use it as a plan view display to view the recording.
- The component descriptions in Appendix C of *MÄK VR-Forces Developer's Guide* have not been updated to reflect changes made in recent releases of VR-Forces. In particular references to port interfaces are obsolete and the component descriptions do not reflect the use of process state repositories. Fixed-wing and rotary-wing components may be significantly incorrect.
- Using large filled polygons, such as minefields, in overlays can degrade performance.
- You cannot load vector data with geocentric databases. Vector data works only with geodetic or UTM databases.

- You cannot give a helicopter a Set Altitude request that places it at less than the starting height value.
- Nested aggregates do not always follow the leader (the superior aggregate).
- Plans stop executing if they encounter a task that they cannot execute. For example, if the plan for a rotary-wing entity includes the Turn To Heading task, which is not a valid task for rotary-wing entities, the plan stops executing.
- If you delete an entity that has a plan and create a new entity with the same name as the deleted entity, the old plan is displayed in the new entity's Edit Plan window. However, the new entity cannot execute that plan.
- Missiles get initialized with their maximum velocity. They do not accelerate to maximum velocity.
- If you assign a plan to an aggregate, remove the leader (via drag/drop in echelon view), retask the leader and run the simulation, the subordinates follow the old leader on its new task.
- Torpedoes ignore a polygon with soiltype water, but the intervisibility fan does not. So it does not show an accurate representation of who can see who.
- When a subsurface entity gets destroyed it jumps to altitude 0.
- It you task a rotary-wing entity to fly to a waypoint on the ground, it will land when it gets to the waypoint. If opposing force trucks get within range of the rotary wing it will fire. Since the rotary-wing entity is at a slight incline, it will fire directly into the ground and explode.
- Under some circumstances, the second entity in an aggregate does not receive the follow task for aggregate movement. The entity does not follow the leader. Other entities follow aggregate number 2.
- If you uninstall a VR-Forces 3.5.1 or earlier using the InstallShield uninstaller, the entity icon fonts get uninstalled. If you have another application that needs them, such as MÄK Plan View Display, you must manually install them. This is not a problem for VR-Forces 3.6 or later.
- Intervisibility works with linear and areal features, but not with point features.
- Occasionally, VR-Forces is unable to load a scenario. When this happens, delete the contents of the *./data/temp* directory and try loading again.
- Setting the heading of a rotary wing entity whose altitude is less than the **starting-height** parameter causes it to rise up to the **starting-height** value. The workaround is to set a very small (non-zero) **starting-height**.
- If a task is interrupted by a trigger that only has set statements, after the set statements are executed, the plan moves to the next task (if there is one) rather than continuing with the task that was interrupted, as it is supposed to. If there is no additional task in the plan, the entity resumes executing the interrupted task.
- The state-repository-min-tick-period and state-repository-min-tick-period-variance parameters are specified in each OPE entry, but are not implemented.

- If you run VR-Forces in DIS, the application ID of the back-end must be between 3000 and 4000. If you specify an application ID outside of this range, the back-end will abort before the exercise connection is created.