MÄK VR-Forces 3.3-ngc Release Notes

This release note contains the following release-specific information for VR-Forces Release 3.3-ngc:

Document ID: VRF-3.3-3-030304

Systems Supported

VR-Forces 3.3-ngc 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

Platform	Compiler
SGI IRIX6.5	MipsPro7.3 C++ compiler (available from SGI)
Linux 7.2	GNU C++ (GCC) 3.0.2
PC with Windows NT 4.0 SP6 /Windows 2000 SP2/XP	Microsoft Visual C++ 6.0, Service Pack 5

Disk Space Requirements

A full installation of VR-Forces requires approximately 380 MB of disk space. Terrain databases use approximately 62 MB and documentation (primarily the class reference) uses 140 MB.

MÄK Product Compatibility

To build VR-Forces applications, you must link with VR-Link 3.8-ngc. If you are building for HLA and want to link with the MÄK RTI, use MÄK RTI 2.0-ngc.

MÄK VR-Forces 3.3-ngc is a parallel release to MÄK Plan View Display 2.3-ngc. it is not compatible with previous releases of the Plan View Display.

Qt Release Compatibility

If you want to do development using the VR-Forces GUI API, you need to use Qt, a cross-platform GUI toolkit. The VR-Forces GUI was built with Qt release 3.0.5. 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.

Online Help

The online help is in HTML format and uses the default browser for your computer. For all online help features to work, you must enable Javascript in your browser.

Network Compatibility

HLA only

VR-Forces 3.3-ngc was built against VR-Link 3.8-ngc and is compliant with:

- RPR-FOM 1.0 and a subset of 2.0 (draft 6 and 14)
- MÄK RTI 2.0-ngc
- DMSO RTI NG 4 and 6.

VR-Forces 3.3-ngc 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 3.3-ngc 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.

Backward Compatibility

VR-Forces 3.3-ngc code is not fully compatible with previous 3.x-ngc releases. Although we have tried to keep API changes to a minimum, programs built with previous releases may require some porting. Scenarios created with release 2.x or later are compatible with release 3.3-ngc. See comments in "Changes to Entity and Object Naming" on page 8 regarding backwards compatibility of scenarios.

The object parameters database is not backward compatible with previous releases. If you have a customized object parameters database, you will have to merge any changes you have made into the new format. See "Backward Compatibility of Object Parameters Database Files" on page 10.

RPR FOM Versions Supported

VR-Forces 3.3-ngc has built-in support for versions 1.0 and 2.0 (draft 6 and 14) of the RPR FOM. It also supports VR-Link's ability to support alternative FOMs through the FOM Mapper. By default, VR-Forces 3.3-ngc 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 (2.0) to the *DtRprFomMapper* constructor and pass the resulting object to the *DtExerciseConn* 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));
```

RTI Support

The MÄK RTI is included on all MÄK product CDs. It is link-compatible with the DMSO RTI NG. VR-Forces 3.3-ngc has been tested with DMSO RTI NG 4.0 and 6.0.

If you use the MÄK RTI, be sure that VR-Forces can find your FED file, and optional *rid.mtl* file, by putting them in the directory from which you are running, or by setting the environment variable RTI_CONFIG to the directory that contains them.

New Features and Updates

This release of MÄK VR-Forces has the following updates and new features:

- Support for checkpointing
- Improved and expanded terrain database support, as follows:
  - Support for importing ESRI shape files
  - Support for multiresolution bitmap paper maps
  - Support for selecting feature data to display
  - New map file parameters
  - The Paper Map Layers dialog box has been renamed and revised
- Updates to the Echelon View
  - Icons in the Echelon view show entity state (healthy or destroyed)
  - The ability to delete entities from Echelon View
  - Ability to drag entities to a new location from the Echelon View
  - Ability to edit membership of aggregates
- *MÄK VR-Forces Developer's Guide* includes documentation of the GUI API
- Additional examples of how to use the GUI API
- Improved loading of scenarios that require multiple back-ends
- Support for multiple ballistic guns on entities
- More robust batch mode (skips bad scenarios in batch file)
- Support for tuning GUI performance
- Support for saving scenarios with relative paths
- Ability to specify scenario parameters when you create a scenario
- Changes to the Simulation API
- Improvements to the fixed-wing entity model
- New naming conventions and support for editing entity names
- Ability to edit the entity label
- The restriction on application numbers for DIS exercises has been eliminated
- Support for copying and pasting plans and printing them

- Support for asynchronous I/O in DIS.

Saving Checkpoints

This release adds support for saving checkpoints. See section 4.7 in *MÄK VR-Forces User's Guide* for details. To support checkpointing, the following data is now written to scenario files:

- Plan state (current stmt cursor, pending trigger list, plan complete status
- Current task (whether issued by plan or from GUI)
- Resources
- Simulation time
- Exercise start time
- Entity velocity, acceleration, and rotational velocity.

The information to be saved in a checkpoint is stored in process state repositories (new objects containing the state associated with the various processes entities model).

- Scenarios (or checkpoints) saved after the start a simulation save all current plan and task information for each entity, such that when the scenario is reloaded:
  - Entities will resume whatever task they were working on at the time of the save (whether an immediate task, or a task in a plan)
  - Entities will remember state information about their task (e.g. which vertex they were heading toward if they were following a route)
  - Entity resources will be set to the same levels as at the time of the save (any changes to resource levels since the start of the scenario are preserved - so if an entity had already spent all of its ammunition resources at the time of the save, it will not have any when scenario is restored).
  - Orientation, velocity, and acceleration at the time of the save are restored
  - Plan state, including the current statement, plan completion state, and pending trigger state are restored

Loading Scenarios that Use Multiple Back-Ends

When you load a scenario that uses multiple back-ends, if any of the required back-ends are not running, VR-Forces prompts you to remap the objects in the scenario. See Section 4.2.2 in *MÄK VR-Forces User's Guide* for details.

If you save a remapped scenario, you are prompted to save the scenario with the original mappings or with the remappings.

The following variables in *vrfGui.mtl* support remapping:

- VRF_autoRemapInCombinedMode
- VRF_autoRemapInSeparateMode

- VRF_separateModeRemapSiteId
- VRF_separateModeRemapAppNum
- VRF_batchModeModeRemapSiteId
- VRF_batchModeModeRemapAppNum.

See Table 2-2 in *MÄK VR-Forces User's Guide* for details.

Automatic remapping is always enabled for batch mode. See section 4.6.3 for details.

GUI Performance Tuning

VR-Forces 3.3-ngc provides feedback about the performance of the GUI display and allows you to set performance parameters.

You can monitor the network and display frame rates in the Frame Rate Monitor. You can also monitor frame rate statistics for each back-end in your session, and you can specify:

- How frequently statistics are reported
- The sample size used to calculate statistics.

See Chapter 11, "Optimizing Performance" in *MÄK VR-Forces User's Guide* for complete documentation. See Table 2-2 in *MÄK VR-Forces User's Guide* for descriptions of the following new MTL parameters that support performance tuning:

- PVD_consecutiveTimesOverThreshold
- PVD_networkTimeThreshold
- PVD_networkFrameRateBuffer
- PVD_slowNetworkFrameOverride
- PVD_tickManagingEnabled.

GUI API Documentation

The VR-Forces GUI API is now fully documented in *MÄK VR-Forces Developer's Guide*. We have added several new examples to support the GUI API documentation.

Saving Scenarios with Relative Paths

You can now save scenarios with relative pathnames or absolute paths. Using relative pathnames in the scenario file makes it easier to use scenarios that were created on a computer with a different directory structure. Paths are saved relative to the parent directory of the working directory. For example, if you run VR-Forces from *C:\vrf3.2-ngc\bin5*, the paths are saved relative to *C:\vrf3.2-ngc*.

If you enable saving of scenarios with relative paths and you save a scenario that was previously saved with absolute paths, the scenario file is written with relative paths except for the database path. You must edit this path by hand to make it relative. If you edit the database path to make it relative, future saves maintain the relative path.

A new MTL variable, **VRF_makePathsRelative**, controls how paths are saved, where 0 means save absolute paths and 1 means save with relative paths. The default is 1.

Specifying Scenario Parameters when You Create A Scenario

The scenario file specifies several parameters that affect the entire scenario. You can now override the default settings and set these parameters when you create a scenario. This feature is implemented in the Choose Terrain dialog box. See section 4.2.1 in *MÄK VR-Forces User's Guide* for details.

Changes to Entity and Object Naming

In previous versions of VR-Forces, an entity's name and echelon ID were essentially the same thing. In this release, an entity's name and echelon ID are separate. The principle effects in the GUI of this change are:

- Entity names are a distinct concept from echelon IDs
- Users can assign meaningful names to objects as long as the names are unique
- VR-Forces manages echelon IDs. Users cannot directly change them.
- You can display names, echelon IDs, or both in the GUI
- When you create a plan, the plan is still associated with an echelon ID, however all the tasks in the plan they require identification of another entity, such as Follow Entity, use the entity name. This means that you can create plans that task entities to interact with non-VR-Forces entities, because non-VR-Forces entities have names, but do not have echelon IDs.

VR-Forces assigns default names to new entities based on their category in the object parameters database, for example, `USAtRW 1`, `USAtRW 2`, and so on. Objects are named based on their object type, for example, `Point 1`, `Point 2`, and so on. You can change the name given to an entity or an object.

A name must be unique. Names have the following length restrictions (because VR-Forces uses the marking text to send the object names over the network):

- Entity - 11 characters
- Aggregates - 10 characters in HLA exercises, 32 characters for DIS.
- Control objects - 255 characters.

VR-Forces assigns each entity an echelon ID based on its position in the force hierarchy. You cannot change an entity's echelon ID. An entity's echelon ID changes if you reorganize it into or out of an aggregate. It can change if you turn on auto-reorganization and its aggregate unit reorganizes.

You can also set an entity's label. Labels are arbitrary strings that you can assign to objects. They do not have to be unique.

Backwards Compatibility of Object Names

When you open a scenario created with a previous version of VR-Forces, VR-Forces will assign a name (based on the site ID and application number) to any entity whose echelon ID string exceeds the allowable number of characters for the entity type under the new naming scheme. If the echelon ID is less than the length allowed for the entity's name, the echelon ID is assigned as the name. For all practical purposes, this means that most individual entities, and HLA aggregates will be renamed.

All entities will retain their previous echelon IDs.

Plans will be updated to replace echelon IDs with names, where necessary. We recommend that after you open a legacy scenario, that you save the scenario and reload it to synchronize plans and entity names.

Editing Aggregate Units

This release adds the ability to edit the membership of aggregate units. You can add members and remove members by dragging them to and from aggregates in the Echelon View. For details, see sections 5.9.3 and 5.9.4 in *MÄK VR-Forces User's Guide*.

Updated Fixed Wing Model

The fixed-wing model has been revised and its behavior is much more stable. The actuator has changed in that control values determine rotational velocity instead of rotational acceleration. The following fixed-wing parameters have changed:

- *DtFixedWingEntityParameters* class, **fixed-wing-entity-param** type:
  - Parameters Removed: **max–AOA**
  - Parameters Added: **max-climb-rate**
- *DtFixedWingActuatorDescriptor* class, **fixed-wing-component-descriptor** type
  - Parameters Removed: **max–yaw-acceleration**, **max-pitch-acceleration**, **max-roll-acceleration**, **flaps-drag-factor**
  - Parameters Added: **check-terrain**

Backward Compatibility of Object Parameters Database Files

If you created a custom object parameters database under a previous version of VR-Forces, you must edit it to use it with VR-Forces 3.3-ngc. We recommend you make a backup copy of the parameter database before making any changes. The following changes affect object parameters database files:

- Subcomponent lists no longer include an entry for the Organization Manager. You should delete these entries.
- The M1A2 entry has a machine gun as a multiple-weapon example only.
- Kegler missiles (kegler-missile-platform entry) have the object type: 1 (2 4 222 1 3 -1 -1).
- The M1A2 has an anti-radar damage entry. Its damage model specification now includes an entry for munition type 2 4 222 1 -1 -1 -1.
- The damage model specification for the generic CIS attack helicopter platform has a Maverick (munition type 2 2 225 1 4 -1 -1) entry.
- Added a new civilian vehicle entry (object type 1 (1 1 -1 27 -1 -1 -1)).
- All of the radio string types specified under entities' local-objects lists have changed as follows:
  - "local-radio" is replaced by "channel-radio"
  - "local-command-radio" is replaced by "command-channel-radio"
- The radio entry has been removed from all remote-objects lists (radios are no longer created for remote entities).
- The category string parameter has been added to the specification for control object entries. These strings are used to generate more intuitive object names for these objects (e.g. "Route 2"). The new category strings are as follows:
  - Point
  - Line
  - Route
  - Area
- If you have added or modified a fixed-wing entries, change your entry to reflect the above changes described in "Updated Fixed Wing Model" on page 9.
- All entries that specified a component type of "automotive-actuator" in their descriptor list, have been changed from a component-descriptor-type of "actuator-component-descriptor" to the more specific type "automotive-component-descriptor". If you have added any new entries that specify an "automotive-actuator", make sure you change the component-descriptor type to the new "automotive-component-descriptor" type.

Improved and Expanded Terrain Database Support

This release supports the features described in the following sections.

Specifying the Display of Feature Data

If a terrain database supports feature data, you can specify which data to display. See section 8.3.2 “Specifying the Drawing Mode”, in *MÄK VR-Forces User’s Guide* for more information.

Support for ESRI Shapefiles

VR-Forces 3.3-ngc adds support for shapefiles. For details, see section 10.4, “Shapefiles”, in *MÄK VR-Forces User’s Guide*. This release includes an example map file that loads shapefiles – *antigua.map*.

Changes to the Paper Map Layers Dialog Box

The Paper Map Layers dialog box has been renamed the Raster Map Layers dialog box. The dialog box now supports changing the draw order of the paper maps. Maps listed at the top of the box are drawn on top of (after) maps listed below them. Select a map name in the box, and use the "Raise" and "Lower" buttons to move them up and down the list. Changes in this dialog box do not take effect until you press Apply or OK.

Multi-Resolution Bitmaps

The Multi-Resolution Paper Map is a new type of paper map that you can specify in a *.map* file. Multi-resolution paper maps let you:

- Divide image files into segments and load only the image file segments that currently need to be displayed on the screen at a particular time. This feature lets you use large, high-resolution images without using up large amounts of memory.
- Specify use of different images at different screen zoom resolutions. For example, you can use lower resolution images for zoomed-out views, and higher resolution images for zoomed-in views.

Instead of specifying an image file in the paper map entry, you specify an MTL file that contains image information. You can generate the MTL file with the ImageSplitter tool or create it by hand.

See section 10.3.5, “Paper Map Records”, in *MÄK VR-Forces User’s Guide* for details about multiresolution bitmap file format. See section 10.5, “Multiresolution Bitmaps” for an explanation of how to use the Image Splitter tool and multiresolution bitmap resolution levels. The example *berkeley.map* demonstrates multi-resolution bitmaps.

MAP File Changes

Map files support the following new parameters:

- **Name** - Specifies a string name for the paper map. This name is used in the Raster Map Layers dialog box. If you do not specify a name, the filename is used in the dialog box.
- **Alpha** - Specifies the alpha mode to be used when drawing a paper map. The values are Blend and Mask. Blend causes the paper map to be drawn using full alpha blending. Mask causes the paper map to be drawn using a simple 1-bit alpha mask, that is, fully transparent, or fully opaque. Mask mode does not look as good as blend, but it is much faster. The default value is Mask. Multi-resolution paper maps do not support blending. These options are only useful if the paper map file has alpha data saved with it.

Support for c7b Databases

You can load CTDB version 7 terrain databases (.c7b), with the following restrictions:

- We can import gridded and TIN data.
- We provide limited support for MES data, representing it as feature data in VR-Forces. We do not import the geometric model data associated with MES. For example, VR-Forces will represent a house imported from MES data as a point feature (it will not import the polygon data representing the structure itself).

Miscellaneous Updates and Changes to the GUI and Application

- VR-Forces has new command line arguments for setting simulation time and exercise start time. They make it possible to add late-starting back-ends to an already running exercise with correct exercise clock parameters.
  - `-j:` exercise start time (seconds)
  - `-k:` simulation time (seconds)
- By default, the exercise start time and simulation time are both 0.0.
- Application IDs are no longer required to be in the range of 3000 - 4000 for DIS.
- VR-Forces has a new civilian vehicle.
- VR-Forces displays a new dialog box when it closes a scenario. The dialog box is displayed while waiting for back-ends to unload their scenarios. It closes automatically when all back-ends have finished closing. If the process seems to be hung, you can click Continue to end the closing process.
- Batch mode is more robust. If VR-Forces cannot load a scenario specified in a batch file, it skips the scenario and continues to the next batch in the file.
- Entity icons in the Echelon view now show if the entity is destroyed with the standard destroyed icon.
- You can now delete entities that you select in the Echelon view.

- The Create menu has a new option - Edit, which opens an Edit dialog box for the selected entity or control object.
- The Entities menu has a new option - Delete, which deletes the selected entities and control objects.
- The option to set or view marking text has been removed. What used to be considered marking text is now put into the object name member variable of each object and a new item, echelonId, has been added to each object. Code that referenced markingText for either the model data or symbols should be updated appropriately as it will no longer compile.
- The Edit Plan and View Plan dialog boxes support printing plans and copying and pasting plan statements from one plan window to an Edit Plan window. See section 9.3, "Editing Plans." in *MÄK VR-Forces User's Guide* for details.
- You can edit an entity's label in the Edit Entity dialog box. See "Changes to Entity and Object Naming" on page 8 for more information.
- When you make changes to the drawing mode on the Terrain View Options dialog box, Drawing Mode tab, you can choose to apply the changes to the Mini Map. To apply changes to the Mini Map, check the Apply Drawing Mode Changes to Mini Map check box.

Simulation API Changes

While we strove to preserve backward compatibility with previous 3.x versions, there are a number of API changes that had to be made. These were necessary to support checkpointing, multiple weapon systems, a revised fixed-wing model, a new echelon organization scheme, and a revised radio model.

Many of the API changes in this release had to be made to support checkpointing. To more completely capture an entity's state as part of a scenario save, we had to make more of the 'process state' maintained by an entity's *DtSimComponents* part of the data saved to the order of battle file. Because the contents of an entity's *DtSimComponents* are not saved in a scenario, we had to move those state member variables out of the *DtSimComponent* classes and into new *DtVrfProcessStateRepository* classes, which are saved as part of an entity's state. The new process state repository classes are designed to hold the state data associated with one or more component models of a given process (for example, movement, execution of a weapon system). If you were directly using a member variable that has moved (for example, out of a *DtSimComponent* and into a corresponding process state repository), then you will have to modify your code. Instead of accessing the old member variable in a *DtSimComponent* class directly, you will have replace this code with calls to the accessors and mutators in the corresponding process state repository class.

The interface to the *DtSimComponent* constructor has changed. The constructor now takes a pointer to the entity it belongs to and a pointer to the component descriptor used to specify a component. This change affects the component factory and all derived component classes. The setEntity() and setComponentDescriptor() member functions are obsolete. They have been retained in the base class and are still called in the Component Manager for backward compatibility. Any functionality previously done in setEntity() has been moved to the constructor or the init() member as appropriate. The following related changes have been made:

- The **acquire-time** parameter has been removed from *DtVrfObjectParameters* (*vrfObjParams.h)*.
- In *DtSimComponent* (*simCmpnt.h*), we added the ability for a component to connect to multiple port interfaces, even if they all have the same name. The *DtTargetDetectionSensor* has been modified to support connecting to multiple port interfaces. This allows this sensor to feed the same target list to as many controllers as may want it.
- In *DtVrfObjectStateRepository* (*vrfObjSR.h*) we removed the current target, target acquired, and user target override members. These parameters have migrated to the weapon PSR used by each weapon subsystem. This enables multiple weapons per platform. This affects the following class interfaces:
  - *balGun.h*
  - *balGunCtlr.h*
  - *baseWeapCtlr.h*
  - *indFireCtlr.h*
  - *indLifeFormAcqr.h*
  - *launchCtlr.h*
  - *turretAcqr.h*
  - *turretAct.h*
  - *turretCntlr.h*
  - *turretScan.h*
  - *weaponAct.h*
- *DtSimComponent* (*simCmpnt.h*) has a postAddComponentsInit() member function that gets called after all of an entity's components have been created and their init() functions called. It provides a convenient place to do any additional initialization of a component that is dependent on the existence of any other components. The implementation of postAddComponentsInit() in the *DtSimComponent* base class is empty. Derived classes can override this function to do any processing that must take place after all components have been created. In particular, we use it to do the lookups of process state repositories for those components that need to obtain pointers to a process state repositories. It can only be done successfully after the component responsible for creating the process state repository has been created, so process state repository lookups are performed there.

- Formation type has moved out of *DtSimSubordinateManager* (*subordMgr.h*) and into the new *DtPseudoAggregateMovementPSR* (*pseudoMovePSR.h*), so it can be saved as part of scenario.
- The myActuatorParametersFound and getAcutatorParameters() members of *DtFixedWingPilotController* (*fixWngPltCtl.h*) and *DtRotaryWingPilotController* (*rotWngPltClt.h*) have been removed from these classes. The concept of a process state repository providing a shared container for process state (shared amongst components) renders these members obsolete. For example, we no longer need to have the *DtFixedWingPilotController* look up the *DtFixedWingActuator* to find out about flight-related parameters. Instead, both components can set/get these parameters in the *DtFixedWingFlightPSR*.

Changes to Weapons Systems

The new *DtVrfWeaponPSR* is a process state repository for sharing data (and reading/writing) data needed by any weapon system. This PSR replaces the entity SR current target and target acquired fields, as well as encapsulating the ammo loading system that was spread among the different weapon controllers. Because an entity can have multiple, uniquely named process state repositories, we can now support multiple weapon systems on a single platform. We were previously limited by the fact that there was only one set of current target and target acquired fields in the entity state repository. This had meant only one weapon system would work.

We have added the *DtMunitionLoader* (*munLoader.h*) class, which encapsulates ammunition loading code that was formerly distributed among various controllers. The *DtWeaponProcessSR* has a munition loader.

Example of Configuring Multiple Weapons on a Platform

We have added a machine gun to the M1A2 entry as an example of configuring an entity with multiple weapon systems (an M1A2 does not have a M16 mounted on it). We extended the target detection files (*M1A2-fgt.det*, *M1A2-ins.det*, *M1A2-ots.det*) to include probability of detection associated with lifeform entities (because the M1A2 example now includes a machine gun).

Other Changes Related to Weapons Systems

- In the simple rotary wing model, the pilot controller attempts to point the RW at the current target. Since platforms may now have multiple targets, this will attempt to align itself to the target acquired by the first weapon if found.
- The Local Entity Set Controller has been modified to provide minimal support for multiple weapons per platform. Since there is no mechanism to indicate which weapon should get the target specified, it is given to the first weapon system found.
- Fire-now behavior for entities with multiple ballistic guns handles requests to fire an entity's weapon by choosing the first weapon it finds on its platform.

- It is not realistic to set a lifeform's posture to weapon-in-fire position through the set data request mechanism. This posture should only be used when the entity fires its weapon. When a back-end receives a set data request specifying the weapon-in-fire posture, it maps that posture to the nearest reasonable posture, weapon-deployed.
- We were using world location instead of entity location when handling detonation interactions of type *DtDetResEntityProximate*.

Changes to the Organization Manager

The old organization manager class (*DtOrganizationManager*) that was a component of each entity has been removed.

A new class (also called *DtOrganizationManager*) is a member of *DtVrfObjectManager* has an API to view the current organization hierarchy, as well as to make any modifications to the hierarchy. See the *MÄK VR-Forces Developer's Guide* for more information of using the new organization manager. All calls to change the organization should go though this new organization manger.

Each organized entity now also has an Organization Controller component. On entities, this component is responsible only for keeping the echelon ID up to date. On pseudo-aggregates, this component sets and tracks echelon Ids of its subordinates, as the v3.2 organization controller did.

DtVrfObjectBaseStateRepository now has an accessor and mutator for the object's echelon ID.

The *DtRadioMessage* class has new members for channel number and flag telling whether the recipient is an entity name, or an entity echelon ID. This flag defaults to entity name for backwards compatibility. Call setReceiverIsEchelonId(DtTrue) to address a message to an echelon ID.

Changes to Radios

The simulation radios used for entities to communicate with each other have been changed. The old *DtSimLocalRadio* class has been replaced with *DtChannelRadio* and *DtSimLocalCommandRadio* has been replaced with *DtCommandChannelRadio*. Remote radios have been removed. These new radios inherit from the same *DtSimBaseRadio* as the old radios did, and work in much the same way as the old radios from the API perspective.

All the radio receiver and radio transmitter classes have been removed, and there is a *DtRadioMessageExecutive* is used directly in the radio to simplify customization of the radios.

Radio Net class (*DtSimRadioNet*) has been removed. The new channel radios send and receive messages addressed to the channel number they are tuned to. Everyone else tuned to the same channel receives the messages, conceptually the same as the old radio nets.

The following classes have been removed in VR-Forces 3.3:

- *DtSimRadioNet* *simRadioNet.h*
- *DtSimRadioReceiver* *radioRcvr.h*
- *DtSimRadioTransmitter* *radioXmitter.h*
- *DtReceiverTransmitter* *rcvrTrans.h*
- *DtCommandRadioTransmitter* *cmdXmitter.h*
- *DtSimLocalRadio* *lclRadio.h*
- *DtLocalRadioReceiver* *lclRadRcvr.h*
- *DtLocalReceiverTransmitter* *lclRcvrTrans.h*
- *DtUnitRadioTransmitter* *unitXmitter.h*
- *DtLocalUnitReceiverTransmitter* *lclUnitRT.h*
- *DtSimLocalCommandRadio* *lclCmdRadio.h*
- *DtLocalCommandReceiverTransmitter* *lclCmdRT.h*
- *DtSimRemoteRadio* *remRadio.h*
- *DtRemoteRadioReceiver* *remRadRcvr.h*
- *DtRemoteReceiverTransmitter* *remRcvrTrans.h*
- *DtRemoteCommandReceiverTransmitter* *remCmdRT.h*
- *DtSimRemoteCommandRadio* *remCmdRadio.h*
- *DtRemoteUnitReceiverTransmitter* *remUnitRT.h*

Miscellaneous Simulation API Changes

- The fixed-wing model now automatically sets its ordered altitude to the altitude of the waypoint, current route vertex, or entity to follow when executing movement tasks. This results in the fixed-wing entity immediately attempting to move to the desired altitude.

Changes to the Remote Control API

The Remote Control API has changed as follows:

- New *DtBackend* class (*backend.h*)
- *DtVrfBackendListener* (*vrfBeListener.h*) - Added a member function to return the number of back-ends currently loaded in a scenario.
- Changes to *vrfController.h* - Added an accessor for *DtVrfBackendListener*. Added a member function to return the number of back-ends currently loaded in a scenario. You can use the *DtBackend* object to see which back-ends have a scenario loaded by inspecting its isLoaded() member function.

Changes to the GUI API

The GUI API has the following changes:

- myTMMapArea no longer exists as a member variable of the *DtTMWindow* class. It must be replaced with a function call to areaMap(). Any code that references this member variable will no longer compile
- The *DtTMWindow* class has been changed to be a subclass of *DtBaseWindow*, so, any creator functions that look like this:

  ```
  static DtTMWindow * create(QWidget* parent, const char* name, WFlags
    flags);
  ```

  must be changed to look like this:

  ```
  static DtBaseWindow * create(QWidget* parent, const char* name, WFlags
    flags);
  ```

  to have the code compile properly.
- The initMapArea of the *DtPvdWindow* class has changed its prototype from:

  ```
  virtual bool initMapArea(const DtTMFactory* f);
  ```

  to:

  ```
  virtual bool initMapArea(DtTMMapArea* area, const DtTMFactory* f);
  ```

  This accounts for the fact that a window can now support multiple map areas.
- The initMiniMap of the *DtPvdWindow* class has changed its prototype from:

  ```
  virtual bool initMiniMap(const DtTMFactory* f);
  ```

  to:

  ```
  virtual bool initMiniMap(const DtTMFactory* f, QWidget* parent = 0);
  ```

Porting MÄK Plan View Display Examples to VR-Forces

Some of the examples supplied with VR-Forces to illustrate use of the GUI API were written for the MÄK Plan View Display (PVD) API. The PVD examples are contained in a workspace called *pvdExamples.dsw*. If you build these examples as they are supplied, they will only allow you to open databases and view entities that are part of an existing simulation. In other words, they just provide PVD features.

The examples that are considered to be PVD examples:

- drawMissileImpact
- unitStatus
- viewControlObjects
- windowsInfoExtension
- xMarksTheSpot.

You can edit the PVD examples to provide full VR-Forces features, as follows:

1. Modify *main.cxx* to reference *DtVrfGuiApplication* and *DtVrfGuiFactory* rather than *DtPvdApplication* and *DtPvdFactory* as follows:

```
#include "vrfGuiFactory.h" // was--> #include "pvdFactory.h"
#include "vrfGuiApplication.h" // was --> #include "pvdApplication.h"
#include "MachineTypes.h"
#include "userWindow.h"
#include "userMapArea.h"
#include "userSymbolMapper.h"
int main(int argc, char* argv[])
{
DtVrfGuiFactory pf; // was--> DtPvdFactory pf;
DtVrfGuiApplication pa; // was --> DtPvdApplication pa(argc, argv);
pf.setMainWindowCreatorFcn(MyPvdUserWindow::create);
if (!pa.init(&pf))
{
return -1;
}
pa.fileNewWindow();
return pa.exec();
}
```

2. Each file that contains an instance of a subclass of a PVD object (for example, *DtPvdMapArea*), that must also be updated to reference the VRF version of the object. For example (from xMarksTheSpot *userMapArea.h*):

```
#include "vrfGuiMapArea.h" // was --> #include "pvdMapArea.h"
class MyPvdUserMapArea : public DtVrfGuiMapArea // was --> class
  MyPvdUserMapArea : public DtPvdMapArea
```

3. In the *.cxx* file that also contains the object being updated, you must change any superclass reference from the pvdObject to the vrfGui representation. For example:

```
void MyPvdUserMapArea::startIntervisibilityCreation(int iMode)
{
stopObjectCreation();
DtVrfGuiMapArea::startIntervisibilityCreation(iMode); // was -->
  DtPvdMapArea::startIntervisibilityCreation(iMode);
}
```

   You do not have to change the class object name from PVD (although you are not required to keep it the same name) in order to modify the application for VR-Forces usage.

4. Repeat this process for all objects that are a subclass of a DtPvd* object. Objects that are not a subclass of DtPvd (such as a the *userDrawer* class in the xMarks-TheSpot example) do not need to be modified.

! You must link with the VR-Forces libraries. You can copy the link lines from any VR-Forces example *.dsp*.

New GUI API Examples

This release include the following new examples:

- unitStatus - Shows how to create a dockable toolbar and fill it with updating entity information.
- viewControlObject - Shows how to use a filter to hide and show control objects on the map display.
- windowsInfoExtension - Shows how to extend an existing user interface element to add additional information, in this case a "Windows X,Y" coordinate to the Information toolbar and Entity Information dialog box.
- xMarksTheSpot - Shows how to add a new type of model data and use it to create a new symbol that can be displayed on the screen.
- fuelDepot - Shows how to add a new type of control point to the front and back ends for creation and assigning tasks, in this case, a fuel depot.
- mergedSet - Shows how to modify the existing set menu by removing two sets (heading and speed) and combining them into a single Set Heading and Speed set data request with a new dialog box.
- scenarioStatistics - Shows how to display statistical information (number of entities, aggregates, control objects) in a dialog box and use Qt connections to have the system notify this dialog box when statistical information being tracked changes.

Documentation Updates

The PDF version of *MÄK VR-Forces User's Guide* is updated to include all new features. Also, note the following clarifications to the printed version of the manual:

- You cannot set the altitude of an air platform to less than the starting height. However, an entity can travel below the starting height as part of a movement task.

Using Sessions

If two different sessions use terrain databases that overlap, note the following possible sources of confusion:

- The front-ends in each session will see entities from the other session that are simulated in the common portions of the terrain. However, the front-end for one session will have no ability to control the entities that are simulated by the other session.
- If you save the scenario in one of these sessions, it only saves the entities simulated in the session. It does not save the entities from the other session, even though they may be visible in a front-end.
- If one of the sessions pauses or rewinds, the front-end in the other sessions will see the entities in the overlapping region pause or return to their origin.

- If the session you are viewing pauses, entities from other sessions will continue to move on the map.

Extent Records

The following is additional explanation of what an extent record is:

Use an ExtentMetafileRecord to specify a coordinate system when there is no GDB file present. You can then overlay a bitmap on this coordinate system using the PixmapPaperMapRecord. There can only be one ExtentMetafileRecord per map file and it is not valid when used in with a GDB file record.

Bug Fixes

This release fixes the following problems:

- When you started a back-end from the command line and specified a terrain database or scenario, it started up in run mode, rather than pause mode. This is fixed and the back-end will only start in run mode if you use the `-r` command line option.
- If you open new VR-Forces windows, the Entity List dialog box created from one of these windows only shows all entities.
- The Skip Task and Abandon Task commands support multiple entity selection.
- If the View Plan window is open and you rewind a scenario, it correctly tracks task completion.
- Closing the VR-Forces GUI closes child windows.
- When you load a scenario all of the entities get displayed.
- If all members of an aggregate get destroyed, the aggregate uses the destroyed symbol. When you load a scenario that contains aggregates, the formation-following order is as expected.
- If you are using TrueType fonts, the symbols for air aggregates are correct.
- When you load a scenario that contains aggregates, the formation following order is as expected – by echelon ID.

Known Problems

VR-Forces Release 3.3-ngc has the following known problems:

- On IRIX, icons may appear discolored or grainy unless color depth is set to 24-bit.
- True Type fonts are not supported in SGI IRIX. Use bitmaps for entity icons.
- If you are using VR-Forces in a DIS exercise you may experience problems with dropped packets and entities may time out. In HLA, you can avoid dropped packets by configuring the MÄK RTI to use asynchronous I/O. In DIS, 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.

- If you are using a DTED database and "Projected False" is specified in the PvDtedMetafileRecord, entities do not execute movement tasks.
- The *world.gdb* file is based on a DTED database. If you use this database, some entities may not execute tasks. Some entities will not display properly.
- 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.
- Surface entities are not configured to fire weapons or be fired upon. You can customize the object parameters database entries to enable this.
- Joysticks do not work correctly with DirectInput.
- The pseudo-aggregate formation controller ignores entities in a formation that are independently tasked when determining if the promotion order needs to be updated. Currently it has no way to determine if remote entities are independently tasked.
- Uninstalling VR-Forces in Windows via InstallShield uninstalls the entity fonts. If you have another application that uses these fonts, such as the MÄK Plan View Display, the correct entity icons would not be displayed. You must manually install the fonts from their location in the other application that needs them.
- A DtedMetafileRecord that does not have a valid File entry hangs vrfGui.
- 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.
- Subsurface entities use incorrect icons.
- It is possible to delete members of an aggregate and have the aggregate entity itself remain in the scenario.
- If you run VR-Forces with the DMSO RTI on Windows in combined mode, if you exit VR-Forces (the GUI), the back-end does not resign from the federation execution. The GUI and back-end console go away, but the vrfGui process does not shut down. You have to go into Task Manager to kill it. If you do not restart the rtiexec after shutting down VR-Forces, restarting VR-Forces after exiting takes much longer than with a fresh rtiexec, and the behavior of VR-Forces may be erratic.
- 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.
- The target detection sensor does not use a process state repository to store sensor state.
- 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 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.
- If you are using TrueType fonts, the symbols for air aggregates are incorrect.