



©Copyright 1999-2010 Telekinesys Research Ltd. (t/a Havok). All rights reserved. ¹

¹ Havok.com and the Havok buzzsaw logo are trademarks of Havok. All other trademarks contained herein are the properties of their respective owners.

This document is protected under copyright law. The contents of this document may not be reproduced or transmitted in any form, in whole or in part, or by any means, mechanical or electronic, without the express written consent of Havok. This document is supplied as a manual for the Havok game dynamics software development kit. Reasonable care has been taken in preparing the information it contains. However, this document may contain omissions, technical inaccuracies, or typographical errors. Havok does not accept any responsibility of any kind for losses due to the use of this document. The information in this document is subject to change without notice.

Contents

1	Havok Migration Guide	3
1.1	Migrating to 2010.2	3
1.1.1	Common	3
1.1.2	Content Tools	4
1.1.3	Physics	5
1.1.4	Animation	6
1.2	Migrating to 2010.1	7
1.2.1	Common	7
1.2.2	Content Tools	12
1.2.3	Physics	12
1.2.4	Animation	15

Chapter 1

Havok Migration Guide

1.1 Migrating to 2010.2

1.1.1 Common

Transition from GCC-XML to LLVM

The GCC-XML header parser has been replaced with a custom parser based on the LLVM compiler project. The new parser should be downloaded, unzipped, and the environment variable `HAVOK_PARSER_ROOT` set to the directory in which it was unzipped. This should be the root of the installation (i.e. the directory containing the `bin/` and `lib/` directories). `reflectionSettings.cache` files from 2010.1 will work as before, and the interface to `generateReflections.py` is unchanged.

The parser will normally work without configuration, however if configuration is required for custom projects, a file named `havokParserConfig.txt` may be created, in the same directory as the parser executable. Each line in this file is inserted into the command line of the parser. The parser command line is essentially that of the LLVM/CLANG compiler. In general, only include paths (`-I/path/to/dir`) and macros (`-DMACRO`) will need to be set.

Files may now be excluded from parsing by adding `//HK_REFLECTION_PARSER_EXCLUDE` anywhere in the file. Sections of a file may be excluded using `#ifdef __HAVOK_PARSER__ / #endif`. (`__GCCXML__` is still supported for compatability). The `addParserExcludeDir` list in `reflectionSettings.cache` can be used to exclude an entire directory from parsing. One of these methods should be used to exclude any code that is not parsed correctly and does not require reflection or memory tracking.

The format of reflected files and classes has not changed.

Version patch changes

Additional patch functions are now available to set the defaults for a class in the patch. This should be used in addition to setting the default in the reflected class, and allows the default to be properly viewed by the patching and reflection system in all cases. Patches Version Verify should report any new patches required, as before. It will now check that the default set in the most recent patch matches that set in the reflected class.

The format of the vector / transform patch macros has changed, HK_PATCH_MEMBER_ADDED_VECTOR4 and HK_PATCH_MEMBER_ADDED_QUATERNION are renamed to HK_PATCH_MEMBER_ADDED_VEC_4, HK_PATCH_MEMBER_ADDED_TRANSFORM becomes HK_PATCH_MEMBER_ADDED_VEC_12 and HK_PATCH_MEMBER_ADDED_QSTRANSFORM is renamed as HK_PATCH_MEMBER_ADDED_VEC_16

C String and pointer members may also be added using the simple patch macros, HK_PATCH_MEMBER_ADDED_CSTRING and HK_PATCH_MEMBER_ADDED_POINTER

hkSerializeUtil, hkHavokSnapshot, hkTagfileWriter interface changes

All of these classes now have Options structures instead of passing boolean options directly. In every case, the default behaviour has not changed.

hkSerializeUtil now has options to:

- save in a text format (hkXmlTagfileWriter)
- save members not normally serialized (for debugging)
- save human readable copies of hex floats
- load only data which does not require versioning

XML Tagfile format created

A new XML file format has been introduced which fixes some fundamental issues with the old XML packfile format.

Floating point values are now written as their big endian hex value by default to avoid loss of precision on round tripping. There are options to use the normal human readable format or to add them as comments. Also, the hex floating point format is independent of locale settings.

Compression Features Added

A lightweight LZ style compressor and stream wrappers have been added. See hkCompression, hkCompressedStreamReader and hkCompressedStreamWriter. Decompression uses no memory and is similar in speed to a naive memcpy. Binary tagfiles typically compress by roughly 40%.

Improved Memory Leak Report

The memory leak report now searches leaked blocks for pointers to other leaked blocks. Thus it can generally give much more accurate reports of the cause of the root leak and suppresses the incidental details. This is particularly noticeable when a leak involves a chain or tree of referenced objects.

1.1.2 Content Tools

There are no major migration issues for Havok Content Tools 2010.2. Please refer to the release notes for details of any minor interface and behavior changes, as well as new features, bug fixes etc.

1.1.3 Physics

hkAabbTree is deprecated

hkAabbTree and the option `hkpWorldCinfo::m_useMultipleTree` which maintains an AABB tree for fast raycasting of inactive objects has been deprecated. Newer APIs provide better performance and memory usage. Other options are KDTree and the new hybrid broad phase.

New Hybrid BroadPhase

A new method to accelerate world queries (ray-cast, linear-cast, etc) is available and can be enabled with settings `hkpWorldCinfo::m_useHybridBroadphase` to **true**. The user guide has more detailed information. See the section "Using hkpHybridBroadphase to improve world queries performance" in the Asynchronous Queries collision detection chapter.

In addition, this new broadphase allow for convex queries, see demo: [Demo/Physics/Api/Collide/Culling](#).

Constraint Stability Improvements

A new solver for constraints based on the `hkpBallSocketConstraintAtom` is available for the ball-and-socket, hinge, limited hinge and ragdoll constraints. The solver is capable of simulating previously unstable configurations (i.e. long chains, pairs of constrained bodies with disproportionate inertias or long force arms) in a more stable manner. It is disabled by default, and provided utility functions (see `hkpConstraintStabilizationUtil`) enable it for all constraints attached to a rigid body / in a physics system / physics world.

For certain configurations, the new solver may still fail to reach a stable solution. To prevent that from happening, one should use either one of the methods provided by the `hkpConstraintStabilizationUtil` or the *Create Constraints* asset export filter UI, and pre-stabilize the simulated system.

HVK-5740	Improve stability of ball-and-socket constraints.
HVK-4682	Create constraints filter should use smart logic to scale inertia tensor / angular damping to make systems stable by default.
HVK-4605	Ball and socket constraint gains energy in a very simple situation.

Per Object Slow Motion Feature

Two new methods, `hkpMotion::getTimeFactor()` and `hkpMotion::setTimeFactor()`, allow for controlling the time scale on a per motion basis.

Collision Detection Fixes

PSI collisions are now processed at every step of an asynchronous simulation (HVK-5701), and not only when the PSI step is reached. This only affects customers who carried out multiple asynchronous steps per PSI step.

Constraint Violated callbacks are now generated for singlethreaded and asynchronous simulations, and not only multithreaded as before (HVK-5687)

The linear cast agent now handles welding correctly. Previously, welded contact normals could cause the agent to return incorrect results (HVK-5642) The normal returned by the agent is still welded if welding

is enabled.

hkpCdPoint.m_contact has been made protected

In order to access the contact point, use `getContact()` and the various `setContact()` methods.

1.1.4 Animation

Cropping of extracted motion has been fixed

The animation system now correctly crops the extracted motion. To fix this required some interface changes to be made. The interfaces affected are not commonly used, but you may encounter them if you are doing something out of the ordinary or if you have written custom subclasses of `hkaAnimation` or `hkaAnimatedReferenceFrame`, in which case you will have to update the interfaces on your custom subclasses.

The virtual method `hkaAnimation::getExtractedMotionDeltaReferenceFrame()` has additional arguments. The old interface was:

```
virtual void hkaAnimation::getExtractedMotionDeltaReferenceFrame( hkReal time, hkReal nextTime, int loops,
    hkQsTransform& deltaMotionOut ) const;
```

The new interface is:

```
virtual void hkaAnimation::getExtractedMotionDeltaReferenceFrame( hkReal time, hkReal nextTime, int loops,
    hkQsTransform& deltaMotionOut, hkReal cropStartAmount, hkReal cropEndAmount ) const;
```

The virtual method `hkaAnimatedReferenceFrame::getDeltaReferenceFrame()` also has additional arguments. The old interface was:

```
virtual void hkaAnimatedReferenceFrame::getDeltaReferenceFrame( hkReal time, hkReal nextTime, int loops,
    hkQsTransform& deltaMotionOut ) const = 0;
```

The new interface is:

```
virtual void hkaAnimatedReferenceFrame::getDeltaReferenceFrame( hkReal time, hkReal nextTime, int loops,
    hkQsTransform& deltaMotionOut, hkReal cropStartAmount, hkReal cropEndAmount ) const = 0;
```

HKA-1372

The cropping feature in the `hkaDefaultAnimationControl` does not correctly crop the extracted motion when looping.

1.2 Migrating to 2010.1

1.2.1 Common

Keycode Change

All Client and Evaluation keycodes have changed for Havok 2010.1. You should have received a new keycode with your download details. If you have not please contact your account manager for a new keycode. PcXS and Ps3XS keycodes have not changed.

Product Feature Deadstripping

It is now easier to strip out unused parts of the SDK without rebuilding the libraries. Product features are only linked in if explicitly requested. This release supports some of the most common targets for removal, more will be added in future releases.

It is now required that you include `hkProductFeatures.cxx` in your project. Typically this replaces your include for `hkBuiltinTypeRegistry.cxx`. Usually you will want to do this in conjunction with your existing keycode include because the feature registration depends on which keycode values are set.

```
#include <Common/Base/keycode.cxx>
#include <Common/Base/Config/hkProductFeatures.cxx>
```

The `hkProductFeatures.cxx` ties together the optional features of the SDK.

- `hkClass` registration
- Versioning patch registration
- Memory tracker registration (see memory improvements section)
- Newly optional features such as the inertia tensor computer, heightfields etc.

By default, all features for your product combination are enabled. To disable features, place the relevant preprocessor macros between the two includes. For example if there is a keycode value for animation, but the application only needs physics we could use:

```
#include <Common/Base/keycode.cxx>
#undef HK_FEATURE_PRODUCT_ANIMATION // was defined by the keycode include
#define HK_EXCLUDE_LIBRARY_hkpVehicle // not using hkpVehicle library
#define HK_EXCLUDE_FEATURE_SerializeDeprecatedPre700 // not using old serialization
#define HK_EXCLUDE_FEATURE_RegisterReflectedClasses // if we register the classes elsewhere (using
    hkBuiltinTypeRegistry.cxx)
#define HK_EXCLUDE_FEATURE_MemoryTracker // not using memory tracker
#include <Common/Base/Config/hkProductFeatures.cxx>
```

See `Common/Base/Config/hkProductFeatures.inl` for a list of switches and descriptions. Some features have been replaced by this process. For example, using the default options, one

no longer needs to call `registerPatches()`, `#include hkBuiltinTypeRegistry.cxx` nor call `hkSerializeDeprecated::initDeprecated()`

Other macros which affect the operation are `HK_CLASSES_FILE` (for a custom class list) and `HK_COMPAT_FILE` to cut down on the number of old versions supported.

As part of this refactoring `hkSerializeDeprecated::initDeprecated()` has been renamed to `hkFeature_serializeDeprecatedPre700()`.

COM-1059	Create a config file that allows customers to dead-strip pieces of code they aren't using
----------	---

Memory Improvements

Garbage Collection

The `hkMemorySystem` interface has been extended for easier freeing of cached memory. `hkMemorySystem::garbageCollect()` will now garbage collect the calling threads local cache before collecting the shared cache. Each thread may call `garbageCollect()` though this may lead to duplication of work as the shared area will be collected several times. There are new methods which allow finer grained control (`garbageCollectThread()` and `garbageCollectShared()`)

COM-973	Garbage collection should call <code>hkThreadMemory::releaseCachedMemory()</code>
---------	---

Solver Buffer Size

`hkMemoryInitUtil::initDefault()` now requires you specify an allocator and default size for the solver buffer. See the 'Solver Buffer Size' of the user guide for more info.

COM-960	Solver buffer size must be set explicitly
---------	---

Memory Tracking and Reporting System

Note:

This is a beta feature.

The memory tracker system is a new system that aims to provide information about memory usage largely automatically. The tracker enables creating a snapshot of memory usage which can then be processed by a variety of reports to provide information about memory usage. The snapshot captures memory allocations, as well as a description of how memory is used. For more details on what the memory tracker system is and how it works, please look at the section in the main Havok Guide.

The key points in using the memory tracker system are as follows..

- You must be using a `hkMemorySystem` that implements `memoryWalk`
 - `hkCheckingMemorySystem` and `hkFreeListMemorySystem` do
 - In demos you can get the checking memory system by giving `c` option on the command line
- You must have a library build compiled with `HK_MEMORY_TRACKER_ENABLE` (currently enabled if `HK_DEBUG` is enabled)
- You must explicitly enable memory tracking. (`hkMemoryInitUtil::initMemoryTracker`)

To try out memory tracking in the Havok demos use the options 'c me' on the command line. 'me' will cause the demo system to output a collection of memory tracker reports once a demo is exited. -c makes the Havok demos use the `hkCheckedMemorySystem` which provides stack traces for all allocations as well as implementing `memoryWalk` a feature required for memory tracking operation. If the demo uses a lot of memory it can take a fair amount of time (perhaps minutes) to dump out all of the reports.

To incorporate memory tracking into your own project a good starting point would be to check out `hkMemoryInitUtil` which has two methods `initMemoryTracker` and `quitMemoryTracker`. The tracker should be initialized just after the memory system and before `hkBaseSystem::init`, otherwise some allocations will be missed. Similarly the tracker should be quit after `hkBaseSystem::quit()` and before the memory system shutdown.

The VDB has been extended to grab a snapshot report in machine friendly plain text which may be analysed offline. A sample tool (`MemoryAnalyzer.py`) is supplied to visualize the data.

To include custom classes in the report, it must contain a memory allocator declaration (`HK_DECLARE_CLASS_ALLOCATOR` or `HK_DECLARE_NONVIRTUAL_CLASS_ALLOCATOR`). Additionally for the most accurate data, information about contained pointers is generated by GccXML parsing. If no such data is available, the blocks are scanned for values which look like pointers.

A variety of different memory reports are available and can be found in the 'Source/Common/Base/Memory/Tracker/Report' directory.

COM-934	Automatic memory tracking
---------	---------------------------

Serialization Improvements

When patching fails (usually due to the default patches not being registered), an informative error is now given.

Arrays of 64-bit values will now be serialized correctly to tagfiles. Previously they were silently truncated to 32-bits when serialized. Any assignment or read to a `hkDataObject` int array that evaluates to a 64-bit data type or value will cause the array to become 64-bit, and sign extension will occur. A warning is output in this case. If the array is only accessed using `int/hkInt32`, it will remain as a 32-bit array.

COM-1017	Patching can fail without a helpful error message
COM-1049	Arrays of 64-bit ints not supported

Reflection Parser

A new reflection parsing system is used for 2010.1 . This uses `GCC-XML` to generate reflected class files and memory tracker definitions. The old script `regenerateXml.py` is replaced by `generateReflections.py` . The key differences are:

- The new system operates on one project at a time, not one file. Reflected classes are output into the `Classes` directory in each project. Old '*Class.cpp' files will be automatically renamed to avoid conflicts.
- It compiles each file using a version of `GCC`. Each header file must therefore contain the required includes and include paths must be configured correctly
- `reflectionSettings.cache` is used to set extra include paths for the build
- A cache of reflected files, called `reflections.db` is stored in each project to speed up rebuilds.

This file and the contents of the `Classes` directory may safely be deleted if a completely clean build is required.







































Installation

See section 'The GCC-XML parser' in the user guide for more detailed installation instructions, however in general:

- Install GCC-XML from a packaged distribution for MAC or Linux builds, or from <http://www.language-binding.net/pygccxml/download.html> for windows.
- Install pygccxml from the above location. Python 2.5 or 2.6 is required (cygwin python will not work correctly)
- Configure `generateReflections.py` to find GCC-XML. This requires editing the file `Tools/Serialize/reflectionDatabase/gccxmlSettings.py`. Normally only setting `HK_GCCXML_PATH` is required.

Both packages must be installed – GCC-XML and pygccxml. The location of the two packages at the time of release, at the above download location is shown below:

Browse Files for C++ Python language bindings

File/Folder Name	Platform	Size	Date ↓	Downloads	Notes/Subscribe
Newest Files					
 pydsc-0.4.zip		9.5 KB	2009-02-15	939	
All Files					
▼  pydsc		52.0 KB	2009-02-15	1,565	 
▶  pydsc-0.4		9.5 KB	2009-02-15	939	 
▶  pydsc-0.2		42.5 KB	2006-08-24	626	 
▶  ctypes code generator		3.0 MB	2009-01-19	373	 
▼  gccxml-setup		5.5 MB	2009-01-07	1,323	 
▶  gccxml-7-JAN-2009-rev1.127		2.7 MB	2009-01-07	1,100	 
▶  gccxml-22-DEC-2008		2.7 MB	2008-12-23	223	 
▼  pygccxml		37.4 MB	2008-10-20	5,714	 
▶  pygccxml-1.0		21.6 MB	2008-10-20	3,850	 
▶  pygccxml-0.9.5		15.7 MB	2008-02-04	1,864	 
▶  pyplusplus		14.3 MB	2008-10-20	4,173	 
▶  pyboost		11.4 MB	2006-06-07	2,352	 

More detailed information about installation and troubleshooting is found in the user guide. This parser also generates memory tracker files for the Havok libraries.

New reflection class layout

The old reflection parser generated one reflected classfile per header file. The new reflection parser generates one file per project, containing all of the reflected classes for that project. This is located in the **Classes** directory at the root of the project. The **hkClass** format has not changed so existing reflected class files do not need to be regenerated, however new or updated reflected classes should be generated using the new reflection system. The single reflection class file will contain all reflection information for the project, so it will conflict with any old classfiles remaining in the project. For this reason, old ***Class.cpp** files that are encountered during the reflection build are automatically renamed with a **.cpp.0** extension. These files should be removed from the project.

Directory parsing rules

When a directory name is passed to **generateReflections.py**, the following rules are used to determine what is parsed:

- If the specified directory contains a **reflectionSettings.cache** file, it and its subdirectories are parsed as a single project
- If **reflectionSettings.cache** files are found in subdirectories of the specified directory, each directory containing a **reflectionSettings.cache** file and its subdirectories are parsed as a single project
- If no **reflectionSettings.cache** files are found, the specified directory and its subdirectories are parsed as a single project. The default include paths are used.

Managed class manifest generation

Managed class manifests, as used by Havok Behavior, are generated using the reflection parser. The parser is run internally by the **genManagedClassManifests.py** script. The parser must be installed and functioning correctly to rebuild **HavokAssembly** using custom managed classes. In particular, a **reflectionSettings.cache** file may be necessary to set custom include directories.

Custom Project Configuration – reflectionSettings.cache

The parser compiles each file and must have enough information to access all required header files. Custom configuration is carried out by creating a file called **reflectionSettings.cache** in the root directory of the project. This file contains a list of settings, in the form of a Python dictionary. There are three configuration values that can be set from this file:

- **addIncludePath** – This is a list of extra directories to add to the search path while searching for header files
- **setPchFile** – Set a precompiled header file. While precompiled headers are not used by the parser, this file is automatically included for each file in the project
- **setPrefix** – Set the prefix to prepend to any generated files. This may be used to distinguish between files from different projects

A sample **reflectionSettings.cache** file is shown below:

```
{'addIncludePath': ['C:/Havok/hk2010_1_0/Source', 'C:/Code/Src'], 'setPchFile': ['stdafx.h'], 'setPrefix': ['hkb']}
```

The file must take this format. The brackets and quotes are all required syntax. Extra includes are added inside the square brackets after `addIncludePath`, as a comma separated list.

Once one `reflectionSettings.cache` file is encountered while searching a directory, the build system assumes that all projects contain this file and it will ignore directories that do not.

Checklist for custom projects

The procedure to convert an old custom project to use the new system for serialized or managed classes is therefore:

- Install and setup GCC-XML and `pygccxml`
- Create a `reflectionSettings.cache` file in the root directory of the custom project, containing any extra includes required to compile correctly. If more than one project is used, create one for each
- Run the parser, either directly as `generateReflections.py` from the command line or via `genManagedClassManifests.py`
- Remove any old classfiles from the project (usually generated files ending in `Class.cpp` . These will have been renamed to avoid conflicts)
- Add the newly generated `Classes/projectNameReflections.cpp` to the project

1.2.2 Content Tools

Support for 3ds Max 2011, Maya 2011 and SoftImage 2011

The Havok Content Tools now support version 2011 (both 32 and 64 bit) of 3ds Max, Maya and SoftImage.

1.2.3 Physics

Added transform per `hkpExtendedMeshShape::TrianglesSubpart`

You can now set an `hkQsTransform` (translation, rotation and scaling) per each `hkpExtendedMeshShape::TrianglesSubpart`, which can be very useful for geometry instancing.

HVK-4156	Add transform to <code>hkpExtendedMeshShape::TrianglesSubpart</code>
----------	--

Inconsistent winding viewer

When exporting a mesh from a modeller you can check the box "Mark Inconsistent Winding Edges" in order to mark and later view the edges and triangles reported as inconsistent welding pairs. There is also a new VDB viewer called "Inconsistent Winding Viewer" which can be useful for analyzing the geometry when meshes are exported with this option turned on.

HVK-5218	Welding viewer should highlight unwelded edges
----------	--

`hkpCompressedMeshShape` improvements

The building process has changed: you have to call `addInstance` at least once for the added geometries and convex pieces in order to correctly compute the AABB of the mesh. Scaling is now supported in the transform (an `hkQsTransform`) when adding geometries and instances. Also enabled collision filtering to work both on CPU and SPU. Another improvement is the fact that degenerate triangles are being filtered now both in the builder and the `getFirstKey` and `getNextKey` functions.

HVK-5314	hkpcCompressedMeshShape: Add a scaling ability like hkpcExtendedMeshShape has
HVK-5354	hkpcCompressedMeshShape does not implement getCollisionFilterInfoImpl
HVK-5510	hkpcCompressedMeshShape can return degenerate triangles

Improvements to the `hkpcTriggerVolume`

The `hkpcTriggerVolume` utility was introduced in 7.1 to demonstrate the use of contact point callbacks in creating a body which senses contacts without having a physical effect. Much work has been done to improve its quality since then and it should now offer a useful alternative to `hkpcPhantoms` and `hkpcPhantomCallbackShapes` as a way of implementing sensors. In particular, `hkpcTriggerVolumes` are sensitive to continuous events.

As part of this work, the `hkpcTriggerVolume`'s interface has been slightly altered. The methods `enterEvent()` and `leaveEvent()`, which could be fired multiple times for the same body, have been replaced by a single method `triggerEventCallback()` which is called at most once for each body. It describes whether the body entered during the frame, left during the frame, or both entered and left during the frame.

Additionally, support for the `hkpcTriggerVolume` has been hard-wired into the `hkpcCharacterProxy`, since the `hkpcCharacterProxy` is responsible for its own integration. To pick up interactions between them, the `hkpcTriggerVolume` fires a variant of the `triggerEventCallback()` which has a `hkpcCharacterProxy` argument.

HVK-5432	hkpcTriggerVolume should order its enter and leave events by when they occur
HVK-5423	The hkpcTriggerVolume can give extra enter and leave events using SIMULATION_TYPE.CONTINUOUS
HVK-5472	hkpcTriggerVolume can fail to raise enter and leave events if a TOI is generated but no PSI contact points
HVK-5467	Trigger Volume should reference count its m-overlappingBodies
HVK-5422	hkpcTriggerVolume can give hkpcCharacterRigidBody support planes
HVK-5415	Visual Debugger (VDB): Improve visualization of hkpcTriggerVolume instances in the VDB
HVK-5418	Demo for comparing the hkpcTriggerVolume, hkpcPhantomCallbackShape and phantoms
HVK-5477	Add unit tests for hkpcTriggerVolume

New approach to the lifetime management of some utility objects

The `hkpcBreakOffPartsUtil` and the `hkpcCollisionCallbackUtil` managed their own memory by adding a reference to themselves and being a `hkpcWorldDeletionListener`. This was inflexible, so we've introduced a new abstraction, `hkpcWorldExtension`, for this kind of object. A `hkpcWorldExtension` is created and then added to the world, at which point the world holds a reference to it, similar to entities. There are changes to the usage of both classes:

- The `hkpcBreakOffPartsUtil` no longer takes the `hkpcWorld` as an argument to its constructor. It should be constructed and then added to the world as a `hkpcWorldExtension`.
- The `hkpcCollisionCallbackUtil` is no longer held in the member `hkpcWorld::m_collisionCallbackUtil`. You can find it using `hkpcWorld::findWorldExtension(HK_WORLD_EXTENSION_COLLISION_CALLBACK)`. To be consistent with the naming used in `hkpcWorldExtension`, its `addCollisionCallbackUtil()` and `removeCollisionCallbackUtil()` methods have been renamed `requireCollisionCallbackUtil()` and `releaseCollisionCallbackUtil()`.

hkpCharacterProxy on SPU

The `hkpCharacterProxy` can now be multithreaded on SPU.

A caveat of using the `hkpCharacterProxy` on SPU is that the callback methods of `hkpCharacterProxyListener` are not fired. Hooks are provided for intercepting the the same information on SPU (see `hkpSpuCharacterProxyUtil`), but issues such as code-size will probably prevent any sophisticated game-code from running there. An alternative is to override `hkpCharacterProxyUtil::handleResults()`.

Creating and running multithreaded jobs is done by using the `hkpCharacterProxyJobUtil`. This has not changed significantly although there are some extra members in the `JobData` struct.

Some internal methods of the `hkpCharacterProxy` class have had their interface changed and many internal classes involved in multithreading the `hkpCharacterProxy` have been changed substantially.

Support for the `hkpTriggerVolume` has also been hard-coded into the `hkpCharacterProxy`.

Changes to hkpBreakOffPartsUtil

The `hkpBreakOffPartsUtil::LimitContactImpulseUtil` was designed to run on SPU. However, when its default behavior was not quite as desired, having it on SPU made customizing it difficult (due, for example, to code size limitations). We have now provided an additional implementation which always runs on CPU to allow easier modification. The `hkpBreakOffPartsUtil` has a factory method, `hkpBreakOffPartsUtil::createLimitContactImpulseUtil()`, which by default creates a `LimitContactImpulseUtilDefault`. If you want a custom implementation, you should override the factory method to create an object of (your subclass of) `LimitContactImpulseUtilCpuOnly`.

As part of this change, `hkpEntity::m_limitContactImpulseUtil` has been renamed `m_limitContactImpulseUtilAndFlag` and is no longer a pointer. You should now use `hkpBreakOffPartsUtil::getLimitContactImpulseUtilPtr()` to obtain a pointer to the utility.

See the section New approach to the lifetime management of some utility objects above for another interface change to the `hkpBreakOffPartsUtil`.

Deprecation of hkpConstrainedSystemFilter

The `hkpConstrainedSystemFilter` has been deprecated and may be removed in a future release. It is recommended that `hkpConstraintCollisionFilter` be used to disable collisions between pairs of constrained bodies.

Added hkpMultithreadedVehicleManager

Note:

This is a beta feature.

The `hkpMultithreadedVehicleManager` has been introduced, supporting multithreaded vehicle execution. The manager generates and executes jobs of type `hkpVehicleIntegrateJob` which calculate the set of impulses to apply to the vehicle and to other bodies, and then applies the impulses afterwards in a deterministic manner. The `hkpVehicleIntegrateJob` is currently **not supported on SPU**. The `hkpVehicleIntegrateJob` offers up to a 40% speed-up versus the `hkpRayCastBatchingManager` or `hkpLinearCastBatchingManager` for a group of vehicles. These only support multithreaded execution of the wheel collision step of vehicle simulation.

HVK-5248	Multithread the Vehicle
----------	-------------------------

1.2.4 Animation

Significant Speed Improvements

New animation decompression and blending modules particularly appropriate for SIMD platforms have been introduced in this release. These new modules consist of the following classes:

- `hkaQuantizedAnimation` - A new, lightweight, compression type optimized for speed of decompression on SIMD platforms.
- `hkaBlender` - A new blend framework optimized for blend speed on SIMD platforms.
- `hkaQuantizedSampleAndCombineJob` - A new job description for sampling and blending which is designed to be general and used on all platforms in single or multithreaded modes.

For optimal performance, `hkaQuantizedAnimation` should only be blended with other instances of `hkaQuantizedAnimation` using the `hkaQuantizedSampleAndCombineJob` job description, or by using an `hkaAnimatedSkeleton` containing only `hkaQuantizedAnimation` instances. Mixing `hkaQuantizedAnimation` instances with other animation types (such as `hkaSplineCompressedAnimation`) will not use the new `hkaBlender` framework and will not provide the best possible performance.

The `hkaQuantizedSampleAndCombineJob` job description contains a flat list of animations to be blended, along with per-job, per-animation and per-bone blending options. The `hkaQuantizedSampleAndCombineJob` is designed to be filled out by users manually, or automatically through the `hkaAnimatedSkeleton` class. Please see *Playback of hkaQuantizedAnimation* section of the *Animation Runtime* chapter of the user guide for further documentation. See also the `Animation\Api\Multithreading\SampleAndBlend\SampleAndBlendMultithreadingDemo` in the Demo Framework for source code examples.

HKA-1339	New SIMD Optimized Sample And Blend Pipeline
----------	--

Important:

`hkaQuantizedAnimation` is available in Havok Behavior. `hkaBlender`, however, is not. This means that `hkaQuantizedAnimations` can be decompressed but cannot be blended using the SIMD blender. Therefore `hkaQuantizedAnimation` inside of Havok Behavior will not yield the same performance gains as when used in Havok Animation when blending many animations. Future work will be done to integrate `hkaQuantizedAnimation` and `hkaBlender` into Havok Behavior.

Deprecation of `hkaWaveletCompressedAnimation` and `hkaDeltaCompressedAnimation` compressed animation types.

`hkaWaveletCompressedAnimation` and `hkaDeltaCompressedAnimation` compressed animation types are now deprecated and may be removed in a future release. Prefer `hkaQuantizedAnimation` for maximum decompression speed, or `hkaSplineCompressedAnimation` for minimum compressed asset size.

HKA-1340	Deprecation of <code>hkaWaveletCompressedAnimation</code> and <code>hkaDeltaCompressedAnimation</code> compressed animation types.
----------	--