Candera V3.0.0

Overview

Release Info

- Release Date: July 10, 2014

Interface Changes

- Candera Engine 3D
- Candera Engine 2D
- Candera Base
- Candera System
- Candera Device
- Candera AssetLoader
- Text Engine

Migration Guide and Backwards Compatibility

In CGI Studio V3.0.0, deprecated interfaces from CGI Studio V2.10.0 can still be used to a large extent, while deprecation functions for interfaces from releases before V2.10.0 have been removed completely.

- When interfaces are used which are marked as **deprecated**, compiler warnings will be generated through the usage of Candera Macro **CANDERA_DEPRECATED_3_0_0**.
- Interface changes in V3.0.0, which are **not backward compatible** to V2.10.0, are explicitely listed in the page below.

Refer to:

- Candera V3.0.0 Migration Guide

Candera Engine 3D

OpenGL ES 3.0 Support

RenderDevice for OpenGL ES 3.0

In order to support OpenGL ES 3.0 a new RenderDevice.cpp has been added, which implements RenderDevice.h with OpenGL ES 3.0 functions. Additionally GlInclude.h, GlTraceMapper and GlTypeMapper have been adapted to support the split of one RenderDevice into several ones for multiple GLES versions. GlTraces are now all displayed under Gles device (instead of Gles20).

For current device packages OpenGL ES 2.0 is still used. If it is supported by a device, usage of OpenGL ES 3.0 can easily be configured in Device.cmake. Simply include Common/OpenGLES30 instead of Common/OpenGLES20 and add the following line to the CMake file.

```
CgiAddDefinitions( -DCGIDEVICE_OPENGLES_30 )
```

Synchronization for OpenGL ES 3.0

The OpenGL ES 3.0 render device uses core OpenGL ES 3.0 fences for synchronization, instead of EGL extension EGL_KHR_fence_sync.

OpenGL ES 3.0 Support for Linux OpenGLAdapter

OpenGLAdapter for Linux hosts now also supports OpenGL ES 3.0 interface (based on GLX and OpenGL 3.0).

OpenGL ES 3.0 ETC2/EAC Texture Compression

OpenGL ES 3.0 introduces ETC2/EAC texture compression as part of the core standard. This enables all OpenGL ES 3.0 capable devices to reduce VRAM and memory bandwidth usage per texture at unnoticeable reduction of image quality, allowing either higher framerates or more and bigger textures used simultaneously.

Candera now supports this feature in the OpenGL ES 3.0 render device. For this purpose Bitmap class has been changed (see Changes in Bitmap Enumerations). The new Bitmap::PixelFormat enumerator has to be used for texture uploading.

See also:

- Import Images for information how to import ETC2/EAC compressed images into SceneComposer.

Cloning Interface

In accordance to basic Unified Cloning Interface, the following changes have been applied to the affected classes in Candera 3D:

- CloneInternal protected interface was removed.
- Clone(Traversing) has become deprecated.
- Copy constructor was implemented or cleaned up.
- Assignment operator was moved to the private section, and its implementation removed.
- Clone() const function was added.

The cloning interface was applied to:

- Node and its descendants: Billboard, Camera, Group, Light, LineList, LodNode, Mesh, MorphingMesh, PlanarShadow, PointSprite, ReflectionCamera, Scene, StereoCamera.
- All other objects that are linked to node and descendants through shared pointers: Appearance (and MultiPassAppearance), Material, RenderMode, Texture, AbstractShaderParamSetter, GenericShaderParamSetter, ShaderParamSetter, Projection, GenericProjection, OrthographicProjection, PerspectiveProjection.
- RenderOrder, as its lifetime is controlled by Scene.

Special considerations:

- General. Clone for shared objects returns a shared pointer to the base shared class.
- General. Descendants of Node implement deprecated "Node* Clone(Traversing)" for backward compatibility.
- Billboard. Bounding box and bounding sphere are copied from the base class, not recomputed.
- Camera. CameraRenderStrategy is not linked to the clone anymore.
- LodNode. Clone contains the same number of levels as the original, but no level information is stored.
- Node. CopyFrom is provided to copy transformations, appearance and other such properties from this node to another one.
- PlanarShadow. Clone loses association to light.
- PlanarShadow. As defined by the Copy constructor (but not the assignment operator, previously used for cloning), the id of the clone is different from the one of the original.
- ReflectionCamera. Clone loses association to source camera.
- Scene. Scene may be cloned. The clone does not receive a render order.
- StereoCamera. ProjectionListener may be copied.

- ShaderParamSetter. Clone does not hold any data since data is handled by the application.
- AssetLoader nodes. They don't support cloning.

Deep Cloning Interface

In accordance to basic Tree Cloning Strategies, the following support is provided for deep cloning in 3D as follows:

- TreeCloner is a specialization of TreeClonerBase.
- DeepNodeCloneStrategy provides a generic node strategy that clones the delegates the cloning of appearance and derived Nodes to specialized strategies.
- DeepAppearanceCloneStrategy provides deep appearance cloning. It clones its constituents by resolving shared instances.
- DeepCompositeGroupCloneStrategy, DeepLodNodeCloneStrategy, DeepStereoCameraCloneStrategy and DeepReflectionCameraCloneStrategy resolve node binding by use of TreeMatch.
- DeepCameraCloneStrategy resolves the sharing of projections.
- DeepSceneCloneStrategy clones the RenderOrder.
- DeepTreeCloner is a wrapper of DeepNodeCloneStrategy that provides an interface similar to TreeCloner.

CompositeGroup

CompositeGroup is a new Node type that is usually created by AssetLoader and which contain:

- a list of AnchorPoints: descendants of the CompositeGroup that can be accessed directly from the CompositeGroup instance.
- a list of Widgets that act on any node within the CompositeGroup subtree.
- a list of AnimationPlayers that act on any node within the CompositeGroup subtree.

Picking Test: Consistent Behavior for deep and flat Picking

In both cases Node::IsPickIntersectingGeometry only returns true for a tested node if it is hit by the intersection ray AND Node::IsEffectiveIntersectionTestEnabled is true.

DeviceObject Listener

DeviceObject now supports adding listeners that are notified before and after the object is uploaded and unloaded and before it is destroyed.

Upload to Multiple Contexts

Device Objects of a Node (and its descendants) can be uploaded to all render target contexts belonging to the cameras the nodes are in scope of.

See also:

> Node::UploadAll(). Furthermore, logical operators for intersection and union of Scope masks have been added to Candera::ScopeMask.

ExternalTextureImage

Added ExternalTextureImage which is a texture image that allows wrapping external created native texture objects.

Car Paint Shaders added

The following reference shaders have been added:

- RefCarbon / RefTransLight1Carbon
  Creates a car paint with typical carbon look.

- RefFlopCarPaint / RefTransLight1FlopCarPaint
  Creates a car paint with shiny, anisotropic two color look.

- RefMetalFlakesCarPaint / RefTransLight1MetalFlakesCarPaint
  Creates a car paint with shiny, sparkling look produced by metal flakes within specular highlight.

- RefFlopMetalFlakesCarPaint / RefTransLight1FlopMetalFlakesCarPaint
  Creates a car paint with shiny, sparkling look produced by metal flakes within specular highlight combined with anisotropic two color look.

Candera Engine 2D

Deterministic Render Order

In the past, RenderNodes with the same depth value have been rendered in an order which was dependent on the point of time the node has been added to the scene. The result was that in SceneComposer the render order could look different as when

loading the scene from the asset on target.

The new implementation ensures that the render order of nodes with the same depth value is defined by the position within the scene tree. Nodes which appear on top of the scene tree (as shown in SceneComposer) are rendered before nodes that are located on the bottom of the scene tree. Therefore the render order is always the same during design on host (SceneComposer) and on target.

Cloning Interface

In accordance to basic Unified Cloning Interface, the following changes have been applied to the affected classes in Candera Engine 2D:

- CloneSelf(CloneOperation) protected interface was removed.
- Clone(TraverseOperation, CloneOperation) has become deprecated.
- Copy constructor lost the optional CloneOperation argument.
- Clone() const function was added.

The cloning interface was applied to:

- Node2D and its descendants: Camera2D, CompositeGroup2D, Group2D, RenderNode, Scene2D.
- Effect2D and Layouter already define the interface appropriately.

Special considerations:

- General. Descendants of Node implement deprecated "Node2D* Clone(Traversing) const" and "Node2D Clone(TraverseOperation, CloneOperation) const" for backward compatibility.
- Camera2D. Camera2DRenderStrategy and RenderTarget2D are not linked to the clone anymore.

- Node2D. Layouter is copied during deep node cloning. Because of this Node2D handles the destruction of Layouter upon disposing the node.
- Scene2D. Scene2D may be cloned. Scene2D render order is not cloned, as it doesn't support different render order bins.

Deep Cloning Interface

In accordance to basic Tree Cloning Strategies, the following support is provided for deep cloning in 2D as follows:

- TreeCloner2D is a specialization for the 2D scene tree of TreeClonerBase.
- DeepNode2DCloneStrategy provides a generic node strategy that clones the delegates the cloning of derived Nodes to specialized strategies.
- DeepCompositeGroup2DCloneStrategy resolves node binding by use of TreeMatch2D.
- DeepRenderNodeCloneStrategy resolves the sharing of effects by use of a map.
- DeepTreeCloner2D is a wrapper of DeepNode2DCloneStrategy that provides an interface similar to TreeCloner2D.

CompositeGroup2D

CompositeGroup is a new Node2D type that is usually created by AssetLoader and which contain:

- a list of AnchorPoints: descendants of the CompositeGroup2D that can be accessed directly from the CompositeGroup2D instance.
- a list of Widgets that act on any node within the CompositeGroup2D subtree.
- a list of AnimationPlayers that act on any node within the CompositeGroup2D subtree.

SharedPointer: Deprecated '...Ptr' types

All effect Ptr types like e.g. BitmapBrushPtr are marked as deprecated. Please use e.g. BitmapBrush::SharedPointer instead.

Deprecated BaseEffect2DPropertySetter

BaseEffect2DPropertySetter and its derivate classes became deprecated. Effect2D objects, as other objects with attached MetaInfo, can already animate their properties using built-in

PropertyMetaInfo AnimationPropertySetters, with an improved performance.

AnimationPropertySetters Namespace Correction

All AnimationPropertySetters residing in `<cgi_studio_candera>/src/Candera/Engine2D/AnimationPropertySetters` have been moved from the *Candera* to the *Candera::Animation* namespace. The following classes are affected by this change:

- AlphaNode2DPropertySetter
- BaseNode2DPropertySetter
- BaseTransformable2DPropertySetter
- BaseTransformable2DRelativePropertySetter
- BaseEffect2DPropertySetter
- Effect2DFloatPropertySetter
- RenderingEnabledNode2DPropertySetter
- Transformable2DRelativeRotatePropertySetter
- Transformable2DRelativeScalePropertySetter
- Transformable2DRelativeScaleXPropertySetter
- Transformable2DRelativeScaleYPropertySetter
- Transformable2DRelativeTranslatePropertySetter
- Transformable2DRelativeTranslateXPropertySetter
- Transformable2DRelativeTranslateYPropertySetter
- Transformable2DRotatePropertySetter
- Transformable2DScalePropertySetter
- Transformable2DScaleXPropertySetter
- Transformable2DScaleYPropertySetter
- Transformable2DTranslatePropertySetter
- Transformable2DTranslateXPropertySetter
- Transformable2DTranslateYPropertySetter

> Moved classes are still accessible via the Candera namespace in CGI-Studio 3.0.0 but these deprecated interfaces will be removed in the next release.

Candera2D Bitmap MipMap Flag

Candera2D internally uses MipMapping for Bitmaps, as long as OpenGL is used to render. Whereas this measure typically improves image quality, MipMaps consume ~1/3 more memory and might bear side effects (interpolation between MipMap levels may produce artifacts). Therefore a MipMapEnabled flag has been added.

See also:

Candera::Bitmap::SetMipMappingEnabled() and Candera::Bitmap::IsMipMappingEnabled().

Camera2D Viewport Transformation functions moved

Viewport transformation functions have been moved from Candera::Camera2D to new class Candera::Math2D. Picking in 2D considers window offset.

Reduced size of members

To reduce heap usage some elements in Candera 2D have been reduced in size:

- The camera sequence number from 32 to 16 bit,
  See also:

  Candera::Camera2D::SetSequenceNumber(Int16 sequenceNumber)
- The bitmap width and height from 32 to 16 bit,
  See also:

  Candera::Bitmap
- The child count of Node2D from 32 to 16 bit,
  See also:

  Candera::Node2D::GetChildCount()

DeviceObject2D listener.

DeviceObject2D now supports adding listeners that are notified before and after the object is uploaded and unloaded and before it is destroyied.

Layout Rectangle

The Layout is no longer done based on bounding rectangle. Instead, Node2D has a new property, layout rectangle. Bouding rectangle is used for picking and dirty area management. Computed

layout and bounding rectangle are based on affect layout and bounding rectangle, respectively.

See also:

Node2D::SetLayoutingRectangle() and Candera::Effect2D::GetLayoutingRectangle().

Candera::Node2D uses new Layout Rectangle instead of Bounding Rectangle for layout.

TextBrush and TextBrushCaches compute their Layout Rectangle based on Text Cursor Position. MaximumSize property of TextBrush has been replaced by CacheArea. Bounding Box and Cache Size have been decoupled from Layout. To retrieve the height of a TextBrush with ConstantHeight use GetLayoutingRectangle, and for ActualHeight use GetBoundingRectangle.

Renderer2DListener

A Renderer2DListener allows to listen to Renderer-Events in Candera2D like in Candera 3D with RendererListener.

Candera Base

Camera Groups

CameraGroup is a container that groups cameras from multiple scenes. CameraGroup has an arbitrary number of 2D and/or 3D cameras from different scenes and it is used by applications to easily access and change camera settings uniformly, e.g. in transitions.

See also:

Texture Render Targets

Unified Cloning Interface

Base classes implement a virtual "Clone() const" method, which is overridden in all children. "Clone() const" builds a new instance of the class using the copy constructor. Copy constructors are protected for use only in derived copy constructors. Copying via assignment operator is prohibited, as it could cause problems, like orphaned nodes, memory leaks or dangling pointers. Assignment operators are thus private.

Cloning is shallow, with only shared pointers and constant character pointers being copied. All other coupling via pointers is broken for the clones.

For more complex cloning see Tree Cloner and Tree Cloning Strategies.

Tree Cloner

TreeClonerBase is provided for cloning trees. The default clone strategy is to call the Clone interface on each node, and build the clone tree with the same node pattern as the original tree.

For deeper cloning, TreeClonerBase supports attaching a TreeCloneStrategy. This

TreeCloneStrategy is typically aware of different types of nodes, and delegates to other strategies.

Tree Cloning Strategies

The deep cloning strategies respect the following rules:

- all shared resources are deep cloned as long as they are not device objects; shared resources in the destination are shared in the same pattern as in the source tree.
- node references are rebuilt as long as source references are found as descendants of the source root; descendants are mapped to the destination tree.

- strategies are designed to support the decorator pattern.
- general strategies delegate to more specific strategies.

Bitmap as SharedPointer

Instances of class Bitmap can only be created by calling static method Bitmap::Create() which returns a SharedPointer.

All interfaces have been changed to pass the shared pointer and the corresponding disposer function has been removed from the interfaces.

Old interface (example):

```
void BitmapImage2D::SetBitmap(Bitmap* bitmap, BitmapDisposerFn disposerFn);
```

New interface (example):

```
void BitmapImage2D::SetBitmap(Bitmap::SharedPointer bitmap);
```

Changes in Bitmap Enumerations

Until now, Bitmap class described the organization of the pixels with the enumerators Bitmap::Format and Bitmap::Type.

With introducing OpenGL ES 3.0 ETC2/EAC Texture Compression, it came up that the bitmap description was not suitable for describing compressed images (which have a format descriptor, but no type descriptor). Another problem was that the Bitmap class allowed invalid combinations of Format and Type, e.g. Bitmap::Format::RgbFormat (3 components) and UnsignedShort4444Type (4 components).

In order to enable compressed bitmaps and to avoid invalid combinations the enumerator Bitmap::PixelFormat has been introduced, here Format and Type have been merged to valid enumeration items, while the ETC2/EAC items have been added.

For this reason Bitmap::Format and Bitmap::Type have become deprecated. Of course it is still possible to use custom formats.

Shared Pointer Concept in Animation Framework

Objects of the following classes can only be created by calling static method Create() which returns a SharedPointer:

- AnimationController
- AnimationTimeDispatcher
- AbstractEasingFunction (and all derived classes)
- InterpolationStrategy (and all derived classes)
- AnimationPropertySetter (and all derived classes)

All interfaces have been changed to pass the shared pointer and the corresponding disposer function has been removed from the interfaces.

Old interface (example):

```
void AnimationPlayer::SetController(AnimationController* controller, DisposerFn disposerFn = 0);
```

New interface (example):

```
void AnimationPlayer::SetController(AnimationController::SharedPointer controller);
```

TreeTraverserBase for 'const' Traversing

Interface of template class TreeTraverserBase has been changed to use only Traverse() and ProcessNode(). TraverseConst() and ProcessConstNode() have been removed. This ensures type safe const correct usage but enforce all const-traverser implementations to be changed. Const traversers have to be defined the following way:

```
class MyTraverser : public TreeTraverserBase<const Node> {
    protected:
        virtual TraverserAction ProcessNode(const Node& node)
        {
        ...
        }
};
```

Candera::Animation Namespace extension

Classes and Types located in `<cgi_studio_candera>/src/Candera/EngineBase/Animation` which formerly belonged to the *Candera* namespace were reassigned to the Candera::Animation namespace.

Affected classes/types:

- AbstractEasingFunction
- AnimationBlendedProperty
- AnimationController
- AnimationGroupPlayer
- AnimationKeyframeSequence
- AnimationPlayer
- AnimationPlayerBase
- AnimationPlayerListener
- AnimationPropertySetter
- AnimationTimeDispatcher
- WorldTimeType
- SequenceTimeType
- BackEaseFunction
- BezierInterpolationStrategy
- BounceEaseFunction
- EaseInterpolationStrategy
- ElasticEaseFunction
- ExponentialEaseFunction
- InterpolationStrategy
- KeyframeSequence
- LinearInterpolationStrategy
- PowerEaseFunction
- SplineInterpolationStrategy
- StepInterpolationStrategy

> Moved classes and types are still accessible via the *Candera* namespace in CGI-Studio 3.0.0 but these deprecated interfaces will be removed in the next release.

CanderaObject becomes DynamicPropertyHost

CanderaObject inherits from DynamicPropertyHost. Objects previously inheriting from DynamicPropertyHost now inherit from CanderaObject. ApplicationData is deprecated. The

DynamicPropertyHost allows to attach multiple application data objects.

Candera System

Property Validation

A new macro CdaValidityTest allows the definition of a validation function for Widget, Effect2D and Gdu properties. In case of failed validation the property will be highlighted in SceneComposer property list and an error message will be displayed.

Candera Device

Automatic Context Resource Pools

Context resource pools are created automatically, and assigned automatically to Device objects. If all objects attached to a display share a context resource pool, the display is associated to the context resource pool. Otherwise, the display has a null context resource pool. The interface for replacing context resource pools on the display is deprecated. So is the interface for manually creating context resource pools.

Layout Rectangle of Effects

Effect2D has an interface to compute the layout rectangle. Layout rectangle is the same as bounding rectangle for non-text effects. For text effects the layout rectangle is given by CursorTextMeasureContext, and bounding rectangle is given by GlyphTextMeasureContext.

ActualHight parameter of TextBrush has been removed. Text is now always constant aligned. Applications need to compensate for this parameter by moving the pivot of the node by the offset of the bounding rectangle.

RenderTarget Auto Clearing

Added auto clear flag (RenderTarget::SetAutoClearEnabled and RenderTarget::IsAutoClearEnabled) to Candera::RenderTarget2D / Candera::RenderTarget3D. This flag, unlike the auto swap, does not alter the clear setting of the Camera/Camera2D, but assures that the RenderTarget is cleared automatically before the first camera renders.

Further changes in detail:

- Candera::Camera2DListener has been enhanced by new methods Candera::Camera2DListener::OnRenderTargetChanged() and Candera::Camera2DListener::OnCameraRenderStrategyChanged().
- Candera::CameraListener has been enhanced by new methods Candera::CameraListener::OnRenderTargetChanged() and Candera::CameraListener::OnCameraRenderStrategyChanged().
- ClearMode is always enabled for Candera::Camera.

- Candera::SharedClearMode and Candera::SharedClearMode2D have been introduced.

See also:

Render Buffer Swapping and Clearing

Warping Orientation

The orientation of the display to use during warping can be set on the Display.

See also:

Display::SetWarpOrientation

Warping Image Bounds

In previous versions the warped image was not restricted to bitmap image bounds. The function Display::SetWarpImageBounds() has to be invoked before function Display::SetWarpMatrix() is called.

GraphicDeviceUnit Name

A name may be attached to a Graphic Device Unit.

See also:

GraphicDeviceUnit::SetName()

Candera AssetLoader

AssetDescriptor

AssetContext class has become deprecated and its functionality is taken by the new AssetDescriptor class.

Old Interface:

```
Candera::DefaultAssetProvider* provider = m_assetProvider;
Candera::AssetContext* assetContext = provider->GetAssetContext();
if (assetContext != 0) {
    for (Int i = 0; i < assetContext->GetAnimationCount(); i++) {
        const Candera::Char* animation = assetContext->GetAnimationName(i);
        //Use animation name
    }
}
```

New Interface in 3.0.0:

```
Candera::DefaultAssetProvider* provider = m_assetProvider;
for (Candera::AssetDescriptor::AssetIdIterator animationIdIterator =  provider->
GetAssetDescriptor().GetAssetIdIterator(AnimationLib
); animationIdIterator.IsValid(); ++animationIdIterator) {
    //use animation AssetId: *animationIdIterator
    //or use animation name: provider->GetNameById(AnimationLib, *animationIdIterator, 0)
}
```

AssetId

AssetId is a unique binary identifier that should be used to retrieve objects from the asset though the AssetProvider interface. AssetIds are supposed to replace the identification of objects by their name/path.

AssetIds should not be hard coded, but referenced through Symbolic Names. For more information see Accessing Items via AssetId. AssetIds can also be retrieved via the AssetDescriptor class iterators as shown in the AssetDescriptor example above.

Old Interface:

```
Candera::DefaultAssetProvider* provider = m_assetProvider;
Bitmap::SharedPointer bitmap = provider->GetBitmap(bitmapName);
```

New Interface in 3.0.0:

```
Candera::DefaultAssetProvider* provider = m_assetProvider;
 //copy AssetId directly from SceneComposer, but better use specified symbolic name or retrieve
AssetId assetId = CDA_LIBRARY_ASSETID(0x0, 0x100);
Bitmap::SharedPointer bitmap = provider->GetBitmapById(assetId);
```

Text Engine

Text Alignment

Enumeration type TextAlignment has been replaced by new enum types HorizontalAlignment and VerticalAlignment.

- HorizontalAlignment: HLeft, HCenter, HRight, HStretch
- VerticalAlignment: VTop, VCenter, VBottom, VStretch

Integrated FreeType 2.5

The following new options are available for configuration from CMake:

- FT_CONFIG_OPTION_NO_ASSEMBLER
- FT_CONFIG_OPTION_SYSTEM_ZLIB
- FT_CONFIG_OPTION_DISABLE_STREAM_SUPPORT
- FT_CONFIG_OPTION_USE_PNG
- FT_CONFIG_OPTION_PIC
- TT_CONFIG_OPTION_SUBPIXEL_HINTING
- TT_CONFIG_OPTION_UNPATENTED_HINTING All these options are disabled by default.

RgbColor deprecated

The class Candera::TextRendering::RgbColor has been deprecated. Candera::Color shall be used instead.

Text Measure Context

TextRenderer::GetBoundingRectangle is deprecated. Instead TextMeasureContext is available to provide similar functionality, in a more configurable manner. GlyphTextMeasureContext is a class provided to emulate the behavior of GetBoundingRectangle for ActualHeight alignment parameter. The rectangle returned by this class may be offset from the text position. To have the same alignment as before when rendering the text, the inverse of this offset needs to be applied. CursorTextMeasureContext is a class provided to emulate the behavior of GetBoundingRectangle for ConstantHeight alignment parameter. The rectangle returned by this class does not usually have an offset and it contains the final advancement of the cursor. From the two rectangles, all

values returned by GetBoundingRectangle may be computed.

Candera V3.0.0 Migration Guide

Following a guide how to uprade application code from V2.10.0 to V3.0.0 Candera Interfaces.

- Migration Guide for Backward Compatible Interface Changes
- Migration Guide for Breaking Interface Changes

Besides interface changes, also the build system has been changed significantly in V3.0.0:

- Candera/FeatStd Build System Changes

Regular Expression Search and Replace Patterns

Migration Guide for Backward Compatible Interface Changes

This page provides an overview about interface changes between Candera V2.10.0 and Candera V3.0.0, which are backward compatible.

Interface Changes with Deprecations

These changes do not require immediate adaptation of existing code. Although deprecated interfaces are still accessible, consider switching to the new interfaces as the deprecated interfaces will be removed with the next release.

Description	Candera V2.10.0	Candera V3.0.0

TextAlignment	TextAlignment
	`Leading`
	`Middle`
	`Trailing`
		HorizontalAlignment
		`HLeft`
		`HCenter`
		`HRight`
		
		VerticalAlignment
		`VTop`
		`VCenter`
		`VBottom`

Effect shared pointer types	[Effect]Ptr e.g. `BitmapBrushPtr`	[Effect]SharedPointer e.g. `BitmapBrush::SharedPointer`

const TreeTraverser	`class MyTraverser : public Tree`
	`protected:`
	`virtual TraverserAction`
	`{`
	`...`
	`}`
	`};`
		`class MyTraverser : public TreeTraverserBas`
		`protected:`
		`virtual TraverserAction ProcessNode`
		`{`
		`...`
		`}`
		`};`

AssetDescriptor	`Candera::DefaultAssetProvider`
	`* provider = m_assetProvider;`
	`Candera::AssetContext`
	`* assetContext = provider->GetA`
	`if (assetContext != 0) {`
	`for (Int i = 0; i < assetCo`
	`const Candera::Char* an`
	`//Use animation name`
	`}`
	`}`
		`Candera::DefaultAssetProvider`
		`* provider = m_assetProvider;`
		`for (`
		`Candera::AssetDescriptor::AssetIdIterator`
		`animationIdIterator =`
		`provider->GetAssetDescriptor().`
		`GetAssetIdIterator(AnimationLib);`
		`animationIdIterator.IsValid(); ++an`
		`//use animation AssetId: *animationIdIt`
		`//or use animation name: provider->GetNa`

AssetId	`Bitmap::SharedPointer bitmap= Base::GetAssetProvider()->GetBitmap("MyModule#`	Assign a Symbol Name "myBmp" to the Bitmap Asset and export headers in SceneComposer as "MyAssets.h".
		`#include "MyAssets.h"`
		`Bitmap::SharedPointer bitmap = Base::GetAss`

Animation namespace	`Candera::InterpolationStrategy*`
	`Candera::LinearInterpolationStr`
	`();`
		`Animation::InterpolationStrategy::SharedPoi`
		`Animation::LinearInterpolationStrategy::Crea`
		`();`

New namespace CgiWidgetLibrary	no namespace required	`using namespace CgiWidgetLibrary;`

3D Shallow cloning with flat traversing	`Node* clone = node->Clone();`
	or
	`Node* clone = node->Clone(`
	`Node::Flat);`
		`Node* clone = node->Clone();`

3D Shallow cloning with deep traversing	`Node* clone = node->Clone(` `Node::Deep);`	`Node* clone = TreeCloner` `().CreateClone(*node);`

2D Shallow cloning with flat traversing	`Node2D* clone = node->Clone();`
	or
	`Node2D* clone = node->Clone(Node2D::Flat);`
	or
	`Node2D* clone = node->Clone(TraverseFlat, CloneShallow);`
		`Node2D* clone = node->Clone();`

Bitmap type/format changes	`UInt8* pixels = GetBitmapDataFr`
	`Bitmap* bitmap = &Bitmap(128, 1`
	`Bitmap::RgbaFormat`
	`, Bitmap::UnsignedByteType,`
	`Bitmap::PackAlignment1, pix`
	`true, true);`
		`UInt8* pixels = GetBitmapDataFromArbitrarySo`
		`Bitmap* bitmap = &Bitmap(128, 128,`
		`Bitmap::RgbaUnsignedBytePixelFormat,`
		`Bitmap::PackAlignment1, pixels, 0,0,`
		`true, true);`

GetBoudingRectangle with ExcludeFinalAdvance, ActualTransversalSize	`bound = textRenderer.GetBouding`
	`Measuri`
		`GlyphTextMeasureContext context;`
		`textRenderer.Render(context, layoutingOptio`
		`bound = context.GetTextRectangle();`

GetBoudingRectangle with IncludeFinalAdvance, ActualTransversalSize	`bound = textRenderer.GetBouding`
	`Measuri`
		`CursorTextMeasureContext cursorContext;`
		`GlyphTextMeasureContext glyphContext;`
		`glyphContext.SetNextContext(&cursorContext)`
		`textRenderer.Render(glyphContext, layouting`
		`glyphBound = glyphContext.GetTextRectangle(`
		`cursorBound = cursorContext.GetTextRectangl`
		`bound = Rectangle(`
		`glyphBound.GetLeft(),`
		`glyphBound.GetTop(),`
		`Math::Maximum(`
		`cursorBound.GetWidth() - glyphBound`
		`glyphBound.GetWidth()),`
		`glyphBound.GetHeight());`

GetBoudingRectangle with ExcludeFinalAdvance, ContantTransversalSize	`bound = textRenderer.GetBouding Measuri`	`CursorTextMeasureContext cursorContext; t GlyphTextMeasureContext glyphContext; e glyphContext.SetNextContext(&cursorContext) textRenderer.Render(glyphContext, layouting glyphBound = glyphContext.GetTextRectangle(cursorBound = cursorContext.GetTextRectangl bound = Rectangle(glyphBound.GetLeft(), cursorBound.GetTop(), glyphBound.GetWidth()), cursorBound.GetHeight());`
GetBoudingRectangle with IncludeFinalAdvance, ContantTransversalSize	`bound = textRenderer.GetBouding Measuri`	`CursorTextMeasureContext cursorContext; t GlyphTextMeasureContext glyphContext; e glyphContext.SetNextContext(&cursorContext) textRenderer.Render(glyphContext, layouting glyphBound = glyphContext.GetTextRectangle(cursorBound = cursorContext.GetTextRectangl bound = Rectangle(glyphBound.GetLeft(), cursorBound.GetTop(), Math::Maximum(cursorBound.GetWidth() - glyphBound glyphBound.GetWidth()), cursorBound.GetHeight());`

Extended Interfaces

These changes extend existing interfaces by adding new functionality and do not require any adaptation of existing code. However, custom implementations may be replaced by these library functions.

Description	Candera V2.10.0	Candera V3.0.0
3D Deep cloning with flat traversing	No previous support.	`Node* clone = node->Clone(); DeepNodeCloneStrategy cloneStr DeepNodeCloneStrategy::Pair pa cloneStrategy.Execute(pair, pa`

3D Deep cloning with deep traversing	No previous support.	`Node* clone = DeepTreeCloner()`
		
		or
		
		`TreeCloner treeCloner;`
		`DeepNodeCloneStrategy cloneStr`
		`treeCloner.SetNodeCloneStrateg`
		`Node* clone = treeCloner.Creat`

Header File Relocations

In CGI Studio 2.x, some functionality had been moved from Candera into FeatStd, keeping redirection macros and header files for backward compatibility. In CGI Studio 3.0.0, redirection code has been removed to a large extent. Following is an overview about macros and header files that will be removed.

Candera/System/Diagnostics Cleanup

> To ensure backward compatibility, macros and files formerly located in Candera/System/Diagnostics can still be accessed in CGI-Studio V3.0.0 . Please be aware that this access will be removed with the next release.

Header files from `Candera/Systems/Diagnostics` which only represented redirections to files located in FeatStd should no longer be used.
Please use respective files in `FeatStd/Diagnostics` and the namespace `Featstd::Diagnostics` instead.

Deprecated	Use instead
Candera/System/Diagnostics/Appender.h	FeatStd/Diagnostics/Appender.h
Candera/System/Diagnostics/CharBuffer.h	FeatStd/Diagnostics/CharBuffer.h
Candera/System/Diagnostics/ConsoleAppender.h	FeatStd/Diagnostics/ConsoleAppender.h
Candera/System/Diagnostics/Debug.h	FeatStd/Diagnostics/Debug.h
Candera/System/Diagnostics/DebuggerOutSystemMemoryStatistic.h	FeatStd/Diagnostics/DebuggerOutSystemMemoryStatistic.h
Candera/System/Diagnostics/ErrorHandling.h	FeatStd/Diagnostics/ErrorHandling.h
Candera/System/Diagnostics/FileAppender.h	FeatStd/Diagnostics/FileAppender.h
Candera/System/Diagnostics/LocationInfo.h	FeatStd/Diagnostics/LocationInfo.h
Candera/System/Diagnostics/LogControl.h	FeatStd/Diagnostics/Log.h
Candera/System/Diagnostics/LogEvent.h	FeatStd/Diagnostics/LogEvent.h
Candera/System/Diagnostics/Logger.h	FeatStd/Diagnostics/Logger.h

Candera/System/Diagnostics/LoggerTyped.h	FeatStd/Diagnostics/LoggerTyped.h
Candera/System/Diagnostics/LogLevel.h	FeatStd/Diagnostics/LogLevel.h
Candera/System/Diagnostics/Measurable.h	FeatStd/Diagnostics/Measurable.h
Candera/System/Diagnostics/SystemMemoryStatistic.h	FeatStd/Diagnostics/SystemMemoryStatistic.h

Similarly, all macros contained in files located in `Candera/System/Diagnostics` which only served as redirection to macros located in FeatStd should not be used any more.

Deprecated	Use instead
CANDERA_DEBUG_BREAK	FEATSTD_DEBUG_BREAK
CANDERA_DEBUG_ASSERT	FEATSTD_DEBUG_ASSERT
CANDERA_DEBUG_FAIL	FEATSTD_DEBUG_FAIL
CANDERA_DEBUG_REENTRANCE_GUARD	FEATSTD_DEBUG_REENTRANCE_GUARD
CANDERA_COMPILETIME_ASSERT	FEATSTD_COMPILETIME_ASSERT
CANDERA_UNUSED_PARAMETER	FEATSTD_UNUSED
CANDERA_PANIC	FEATSTD_PANIC
CANDERA_PANIC_IF	FEATSTD_PANIC_IF
CANDERA_LOG_FUNC	FEATSTD_LOG_FUNC
CANDERA_LOG_LOCATION	FEATSTD_LOG_LOCATION
CANDERA_LOG_SET_REALM	FEATSTD_LOG_SET_REALM
CANDERA_LOG_DEBUG	FEATSTD_LOG_DEBUG
CANDERA_LOG_INFO	FEATSTD_LOG_INFO
CANDERA_LOG_WARN	FEATSTD_LOG_WARN
CANDERA_LOG_ERROR	FEATSTD_LOG_ERROR
CANDERA_LOG_FATAL	FEATSTD_LOG_FATAL
CANDERA_LOG_REALM	FEATSTD_LOG_REALM
CANDERA_LOG	FEATSTD_LOG

Migration Guide for Breaking Interface Changes

These changes are incompatible with earlier versions of Candera and will require immediate adaptation of existing code.

Interface Changes without Deprecations

Description	Candera V2.10.0	Candera V3.0.0

Shared pointers in animation framework	`AnimationController* animController`
	`AnimationController::Create();`
	`AnimationTimeDispatcher* animDispatc`
	`AnimationTimeDispatcher::Create();`
	`InterpolationStrategy* interpolation`
	`SplineInterpolationStrategy::Create();`
	`BackEaseFunction backEase;`
		`AnimationController::SharedPointer animCo`
		`AnimationController::Create();`
		`AnimationTimeDispatcher::SharedPointer ar`
		`AnimationTimeDispatcher::Create();`
		`InterpolationStrategy::SharedPointer inte`
		`SplineInterpolationStrategy::Create();`
		`BackEaseFunction::SharedPointer backEase`
		`BackEaseFunction::Create();`

Shared pointers and Create methods for animation property setters instead of constructors	e.g.
	`Candera::Transformable2DTranslateYPr`
	`Candera::MaterialPropertySetter mate`
		e.g.
		`Candera::Transformable2DTranslateYPropert`
		`Transformable2DTranslateYPropertySetter::`
		`();`
		`Candera::MaterialPropertySetter::SharedPo`
		`MaterialPropertySetter::Create();`

Shared pointer destruction	`Bitmap* bmp = CANDERA_NEW(...);`
	`CANDERA_DELETE(bmp);`
	`bmp = 0;`
		SharedPointers may not be manually disposed, freed or deleted. Additionally SharedPointers cannot be set to NULL in the same way as pointers. According code has to be removed:
		`Bitmap::SharedPointer bmp =`
		`Bitmap::Create(...);`
		`// no manual destruction`
		`// no need to set to zero / NULL`

Bitmap as SharedPointer	`Bitmap bitmap(...);`	`Bitmap::SharedPointer bitmap =` `Bitmap::Create(...);`

BitmapTextureImage, CubeMapTextureImage, BitmapImage2D setters take SharedPointer to Bitmap and no disposer function	`BitmapTextureImage textureImage =`
	`BitmapTextureImage::Create();`
	`Bitmap bitmap = CANDERA_NEW(Bitmap)(`
	`textureImage->SetBitmap(bitmap, 0);`
		`BitmapTextureImage textureImage =`
		`BitmapTextureImage::Create();`
		`Bitmap::SharedPointer bitmap =`
		`Bitmap::Create(...);`
		`textureImage->SetBitmap(bitmap);`

Header File Relocations

In CGI Studio 2.x, some functionality had been moved from Candera into FeatStd, keeping redirection macros and header files for backward compatibility. In CGI Studio 3.0.0, redirection code has been removed to a large extent. Following is an overview about removed macros and header files.

CanderaPlatform/OS Cleanup

Removed	Use instead
CanderaPlatform/OS/FixedPoint	FeatStd/Platform/FixedPoint
CanderaPlatform/OS/FractionNumberMath.h	FeatStd/Platform/FractionNumberMath.h
CanderaPlatform/OS/FractionNumber.h	FeatStd/Platform/FractionNumber.h
CanderaPlatform/OS/CodePointIterator.h	FeatStd/Platform/CodePointIterator.h

Candera TimePlatform

Removed	Use instead
Candera/Time.h	CanderaPlatform/OS/TimePlatform.h

Candera/FeatStd Build System Changes

Library Changes

Candera base feature libraries have been joined into the new common library called **Candera.lib**. The following libraries from CGI-Studio V2.x no longer exist separately:

- CanderaEngineBase.lib
- CanderaEngine2D.lib
- CanderaEngine3D.lib
- CanderaGlobalization.lib
- CanderaSystem.lib
- CanderaOsWin32.lib
- CanderaWidget.lib

CMake Switch Name Modification

The names of almost all CMake switches used in CGI-Studio V2.x have been revised to improve effect and feature association. The feature name now reflects, to which component the feature belongs to.

> Please be aware, that even though the CMake Switch names and preprocessor defines from CGI-Studio V2.x still work in CGI-Studio V3.0.0, this backward compatibility is going to be removed with the next release.

Candera CMake Switch Changes

- For detailed information on Candera configuration please see Application Development > Build System Setup > Building Candera.
- For Candera compiler defines refer to Candera Configuration.

V2.10.0	V3.0.0
CGIFEATURE_ENABLE_CANDERA_2D	CANDERA_2D_ENABLED
CGIFEATURE_ENABLE_CANDERA_3D	CANDERA_3D_ENABLED
CGIFEATURE_VRAM_ALLOCATOR_CHECK	CANDERA_VRAM_ALLOCATOR_CHECK_ENABLED
CGIFEATURE_VRAM_ALLOCATOR_STATISTICS	CANDERA_VRAM_ALLOCATOR_STATISTICS_ENABLED
CGIFEATURE_ENABLE_RENDER_STATE_CACHING	CANDERA_RENDER_STATE_CACHING_ENABLED
CGIFEATURE_ENABLE_GLOBALIZATION	CANDERA_GLOBALIZATION_ENABLED
CGIFEATURE_ENABLE_ASSET_DECOMPRESSION	CANDERA_ASSET_DECOMPRESSION_ENABLED
CGIFEATURE_TEXTSHAPER	CANDERA_TEXT_SHAPER
CGIFEATURE_ENABLE_BIDIRECTIONAL_TEXT	CANDERA_BIDIRECTIONAL_TEXT_ENABLED
CGIFEATURE_ENABLE_MULTILINE_TEXT	CANDERA_MULTILINE_TEXT_ENABLED
CGIFEATURE_ENABLE_VIDEO_MEMORY_STATISTIC FEATSTD_ENABLE_VIDEO_MEMORY_STATISTIC	CANDERA_VIDEO_MEMORY_STATISTIC_ENABLED
CGIUNITTEST_ENABLED	CANDERA_UNITTEST_ENABLED
CGIUNITTEST_INCLUDE_ISOLATED	CANDERA_UNITTEST_INCLUDE_ISOLATED
CGIDEVICE_ENABLE_4BIT_GLYPH_CACHE	CANDERA_4BIT_GLYPH_CACHE_ENABLED
CGIFEATURE_ENABLE_LOG	FEATSTD_LOG_ENABLED
CGIFEATURE_ENABLE_MEMORYPOOL	FEATSTD_MEMORYPOOL_ENABLED
CGIFEATURE_ENABLE_TEXTENGINE_MEMORYPOOL	CANDERA_TEXTENGINE_MEMORYPOOL_ENABLED
CGIFEATURE_ENABLE_ASSETLOADER_MEMORYPOOL	CANDERA_ASSETLOADER_MEMORYPOOL_ENABLED
CGIFEATURE_ENABLE_MEMORY_STATISTIC	FEATSTD_SYSTEM_MEMORY_STATISTIC_ENABLED
CGIFEATURE_SYSTEM_MEMORY_STATISTIC_FILE_AND_LINE _TRACKING	FEATSTD_SYSTEM_MEMORY_STATISTIC_FILE _AND_LINE_TRACKING_ENABLED
CGIFEATURE_FRACTIONAL_NUMBER_TYPE	FEATSTD_FRACTIONAL_NUMBER_TYPE
CGIFEATURE_ENABLE_MONITOR	FEATSTD_MONITOR_ENABLED

CGISTUDIO_BUILD_FREETYPE	Has been removed, Freetype will be included automatically.

FeatStd CMake Switch Changes

- Refer to the FeatStd 3.0.0 CMake Changelog to learn about FeatStd V3.0.0 CMake switch modifications.
- For FeatStd compiler defines refer to FeatStd Configuration.

Regular Expression Search and Replace Patterns

The following Regular Expression Search & Replace patterns may be used to quickly apply changes to existing source code. They aim at fixing many (breaking) changes introduced with the new version of Candera. However, as simple text replacement with regular expressions is used, corrections which resulted from applying these patterns may be incorrect in some situations. Therefore, the patterns are just provided as-is in the hope that they may be useful for code migration.

To apply the patterns Notepad++ or any other compatible Regular Expression Search & Replace tool can be used. When using Notepad++ please make sure to select the search method "Regular expression" and check "Match case".

Patterns for Fixing Breaking Changes

Use case	Search pattern	Replace pattern
Declarations of AnimationPlayer pointer	(\\S*)SharedPointer<AnimationPlayer>	AnimationPlayer::SharedPointer
Declarations of AnimationGroupPlayer pointer	(\\S*)SharedPointer<AnimationGroupPlayer>	AnimationGroupPlayer::SharedPointer
Declarations of AnimationTimeDispatcher pointer variables	((\\W)|(const\\s+))(Candera::)?AnimationTimeDispatcher((\\s*)*)(\\s*?[\\w:]+)(\\s*?=\\s*?(NULL|0))?	$2$4AnimationTimeDispatcher::SharedPointer$6$7
Declarations of AnimationController pointer variables	((\\W)|(const\\s+))(Candera::)?AnimationController((\\s*)*)(\\s*?[\\w:]+)(\\s*?=\\s*?(NULL|0))?	$2$4AnimationController::SharedPointer$6$7
Declarations of InterpolationStrategy pointer variables	((\\W)|(const\\s+))(Candera::)?InterpolationStrategy ((\\s*)*)(\\s*?[\\w:]+)(\\s*?=\\s*?(NULL|0))?	$2$4InterpolationStrategy::SharedPointer$6$7

Calls to AnimationKeyframeSequence::SetInterpolationStrategy()	SetInterpolationStrategy\\(\\s*?&?\\s*?([\\w:\\(\\)]+),.*?\\)	SetInterpolationStrategy \\($1\\)
Calls to AnimationPlayer::SetController()	SetController\\(\\s*?&?\\s*?([\\w:\\(\\)]+),.*?\\)	SetController\\($1\\)
Declarations of Animation PropertySetter pointer variables	((\\W)|(const\\s+))(Candera::)?(\\w*)PropertySetter((\\s*)*?)(\\s*?\\w+)(\\s*?=\\s*?(NULL|0))?	$2$4$5PropertySetter::SharedPointer$7$8
Creation of Animation PropertySetter objects	(CANDERA_NEW|FEATSTD_NEW)(\\s*?\\(\\s*?)(Candera::)?([\\w:]*)PropertySetter(\\s*\\))	$3$4PropertySetter::Create\\(\\)
Calls to AnimationBlendedProperty::SetAnimationPropertySetter()	SetAnimationPropertySetter\\(\\s*?&?\\s*?([\\w:\\(\\)]+)\\)	SetAnimationPropertySetter\\($1\\)
Forward declarations of classes for pointer declarations which are now SharedPointer instances	(namespace Candera.*\\{.*)([\\r\\n\\s\\w;]*)(class AnimationController;.*[\\r\\n])	#include <Candera/EngineBase/Animation/AnimationController.h>\\n\\n$1
Forward declarations of classes for pointer declarations which are now SharedPointer instances	(namespace Candera.*\\{.*)([\\r\\n\\s\\w;]*)(class AnimationTimeDispatcher;.*[\\r\\n])	#include <Candera/EngineBase/Animation/AnimationTimeDispatcher.h>\\n\\n$1
Declarations of Bitmap pointer variables	((\\W)|(const\\s+))(Candera::)?Bitmap((\\s*)*)(\\s*?\\w+)(\\s*?=\\s*?(NULL|0))?	$2$4Bitmap::SharedPointer$6$7
Forward declarations of classes for pointer declarations which are now SharedPointer instances.	(namespace Candera.*\\{.*)([\\r\\n\\s\\w;]*)(class Bitmap;.*[\\r\\n])	#include <Candera/EngineBase/Common/Bitmap.h>\\n\\n$1
Calls to e.g. BitmapImage2D::SetBitmap()	SetBitmap\\(\\s*?&?\\s*?([\\w:\\(\\)]+),.*?\\)	SetBitmap\\($1\\)
Calls to CubeMapTextureImage::SetBitmapFace()	SetBitmapFace\\(\\s*?&?\\s*?([\\w:\\(\\)]+),(.*?),.*?\\)	SetBitmapFace\\($1,$2\\)
Use CanderaPlatform/OS/TimePlatform.h instead of Candera/Time.h	(#include <Candera\\/Time\\.h>)	#include <CanderaPlatform\\/OS\\/TimePlatform\\.h>
Use CanderaPlatform/OS/TimePlatform.h instead of Candera/Time.h	([\\s=\\n\\(<])Time::	$1TimePlatform::

Use headers in FeatStd, that were previously located in CanderaPlatform/OS	#include <CanderaPlatform\\/OS\\/Diagnostics\\/(FixedPoint(.*)|FractionNumberMath.h|FractionNumber.h|CodePointIterator.h)>	#include <FeatStd\\/Platform\\/$1.h>

Patterns for fixing backward compatible changes

Use case	Search pattern	Replace pattern
Usage of EaseFunction qualifier	([^(\\w|(Animation::)|\\/|(enum)|(class))]|\\(|(Candera::)|\\s)((Back|Bounce|Elastic|Power|Exponential)EaseFunction)	$1Animation::$3
Usage of InterpolationStrategy qualifier	([^(\\w|(Animation::)|\\/|(enum)|(class))]|\\(|(Candera::)|\\s)((Bezier|Ease|Linear|Spline|Step)?InterpolationStrategy)	$1Animation::$3
Usage of Animation core classes qualifier	([^(\\w|(Animation::)|\\/|(enum)|(class))]|\\(|(Candera::)|SharedPointer<|\\s)(Animation(BlendedProperty|Controller|GroupPlayer|KeyframeSequence|Player|PlayerBase|PlayerListener|PropertySetter|TimeDispatcher|TimeType))(\\W)	$1Animation::$3$5
Usage of Animation core classes qualifier	([^(\\w|(Animation::)|\\/|(enum)|(class))]|\\(|(Candera::)|\\s)(AbstractEasingFunction|CameraGroupAnimationPropertySetter|KeyframeSequence|SequenceTimeType)	$1Animation::$3
New namespace CgiWidgetLibrary	public\\s+(Listener)\\s*?<	public CgiWidgetLibrary::$1<
Removed indirection from Candera to FeatStd for Logging	CANDERA_DEBUG_(\\w*)\\(	FEATSTD_DEBUG_$1\\(
Removed indirection from Candera to FeatStd for Logging	CANDERA_LOG(\\w*)\\(	FEATSTD_LOG$1\\(
Removed indirection from Candera to FeatStd for Logging	([^:\\/\\w]|\\s)(Log(Control|Level))	$1FeatStd::Diagnostics::$2
Removed indirection from Candera to FeatStd	CANDERA_PANIC(\\w*)\\(	FEATSTD_PANIC$1\\(
Removed indirection from Candera to FeatStd	CANDERA_COMPILETIME_ASSERT	FEATSTD_COMPILETIME_ASSERT
Removed indirection from Candera to FeatStd	CANDERA_UNUSED_PARAMETER	FEATSTD_UNUSED

Removed indirection from Candera to FeatStd	#include <Candera\\/Systems\\/Diagnostics\\/(Appender|CharBuffer|ConsoleAppender|Debug|DebuggerOutSystemmemoryStatistic|ErrorHandling|FileAppender|LocationInfo|Measurable|SystemMemoryStatistic|Log(ger(Typed)?|Event|Level|))(.h|Control.h)>	#include <FeatStd\\/Diagnostics\\/$1.h>
Removed indirection from Candera to FeatStd	#include <FeatStd\\/Diagnostics\\/LogRealm.h>	#include <FeatStd\\/Diagnostics\\/Log.h>