BertecKinamotoRuntime.Core

Bertec.BuildServicesInterface

Interface for build service operations such as building projects, setting bundle versions, and generating scene info.

void BuildProject(string[] , string , string , string)

Builds the project with the specified scenes and output parameters.

scenes : Array of scene paths to include in the build.
outputFolder : The output folder for the build artifacts.
outputName : The output file name.
outputPackageName : The output package name.

Bertec.BuildServicesInterface.SetPlayerBundleVersionFromBuldnum()

Sets the player bundle version from the build number.

Bertec.BuildServicesInterface.GenerateSceneInfoJson()

Generates a JSON file containing scene information.

void MakeGenerated(string)

Generates required files in the specified main project assets folder.

mainProjectAssets : The main project assets folder path.

Bertec.BuildServices

Provides static methods and properties for build services, including project building and scene info generation.

Bertec.BuildServices._BuildType

Types of build supported by the build services.

Bertec.BuildServices._BuildType.CustomerScene

Customer scene build type.

Bertec.BuildServices.BuildOutputFolder

The output folder for the build artifacts (e.g., exe, dll).

Bertec.BuildServices.BuildOutputFile

The output file for the build (e.g., .apk or .wpk file). Blank if building loader.

Bertec.BuildServices.SetPlayerBundleVersionFromBuldnum()

Sets the player bundle version from the build number.

Bertec.BuildServices.GenerateSceneInfoJson()

Generates a JSON file containing scene information.

void MakeGenerated(string)

Generates required files in the specified main project assets folder.

mainProjectAssets : The main project assets folder path; optional, defaults to Assets.

void BuildProject(string[] , string , string , string)

Builds the project with the specified scenes and output parameters.

scenes : Array of scene paths to include in the build.
outputFolder : The output folder for the build artifacts.
outputName : The output file name.
outputPackageName : The output package name. Optional.

Bertec.CameraRotationAxis

Specifies the axis of camera rotation.

Bertec.CameraPositionAxis

Specifies the axis of camera position.

Bertec.CameraDisplayModeStatus

Tracks and manages the current camera display mode (HMD, FlatPanel, Dome, etc).

Bertec.CameraDisplayModeStatus.Mode

The current display mode.

Bertec.CameraContainer_Impl

Implements the main logic for managing the camera container, including camera background color, position, and rotation. Handles initialization, state tracking, and provides utility methods for camera manipulation in both HMD and dome projection modes.

Note: The camera container will check if the scene is running on HMD (android) platform and if so, try to turn off the main camera and turn on the Pico stack
for proper rendering. Having both the main camera and the pico sdk running at the same

time will kill performance.
If not HMD, then it assumed to be the Windows player which supports the Dome Projection mode via the UseProjection command line parm.

Bertec.CameraContainer_Impl.DomeProjectionProvider

Gets or sets the dome projection provider for this camera container.

Bertec.CameraContainer_Impl.DOME_IDLE_BACKGROUND_COLOR

The default background color used when the dome is idle.

Bertec.CameraContainer_Impl.WasUsingSkybox

Indicates whether the camera was previously using the Skybox clear flag.

Bertec.CameraContainer_Impl.PREV_COLOR

Stores the previous background color of the camera (needed for scenes like Starfield and used when returning from idle)

Note: Need to know previous background color of the Camera, especially for scenes like Starfield,
because background color can be changed and with returning from idle, it needs to keep the state

Bertec.CameraContainer_Impl.MainCameraAwake

Event triggered when the main camera awakens.

cameraContainer : camera container instance.

Bertec.CameraContainer_Impl.MainCameraStart

Event triggered when the main camera starts.

cameraContainer : camera container instance.

Bertec.CameraContainer_Impl.CameraBackgroundColorChanged

Event triggered when the camera background color changes.

newColor : new background color.
revertClearFlagsToSkybox : if true, reverts the skybox clear flags.

Bertec.CameraContainer_Impl.Instance

The singleton instance of the camera container.

Bertec.CameraContainer_Impl.OwnerContainer

The MonoBehaviour that owns this camera container.

Bertec.CameraContainer_Impl.MainCamera

The main Unity camera managed by this container.

Bertec.CameraContainer_Impl(UnityEngine.MonoBehaviour, UnityEngine.Camera, UnityEngine.GameObject)()

Initializes a new instance of the class.

ownerContainer : The MonoBehaviour that owns this container.
mainCamera : The main camera to manage.
mainContainer : The main container GameObject for rotation.

void ChangeCameraBackgroundColor(UnityEngine.Color , bool)

Changes the background color of the main camera.

color : The new background color.
revertClearFlagsToSkybox : If true, reverts the camera's clear flags to Skybox; otherwise, uses Color.

Bertec.CameraContainer_Impl.RevertCameraBackgroundColor()

Reverts the camera background color to the previous value and clear flags.

Bertec.CameraContainer_Impl.Start()

Initializes the camera container, sets the base position, and subscribes to camera events. Called from Bertec.CameraContainer.Start

Bertec.CameraContainer_Impl.OnDestroy()

Cleans up event subscriptions and instance tracking on destruction. Called from Bertec.CameraContainer.OnDestroy

Bertec.CameraContainer_Impl.ResetCameraRotation()

Resets the camera rotation smoothly to the default orientation.

Bertec.CameraContainer_Impl.ResetRotationSmoothly()

Coroutine to smoothly reset the camera rotation to zero over one second.

void SetCameraRotation(Bertec.SetCameraRotationAngle)

Sets the camera rotation for a specific axis.

rotvals : The rotation axis and angle to set.

void ResetCameraPosition(Bertec.ResetCameraPositionAxis)

Resets the camera position for a specific axis or all axes.

axis : The axis to reset.

void SetCameraPosition(Bertec.SetCameraPositionValue)

Sets the camera position for a specific axis or all axes, enforcing camera limits.

axispos : The axis and position value to set.

void EnforceCameraLimitX(float)

Clamps the X position value to the allowed camera limits.

val : The value to clamp.

void EnforceCameraLimitY(float)

Clamps the Y position value to the allowed camera limits.

val : The value to clamp.

void EnforceCameraLimitZ(float)

Clamps the Z position value to the allowed camera limits.

val : The value to clamp.

Bertec.CameraContainerCallbacks

Provides static callbacks and event registration for camera container RPC commands and state changes. Handles initialization and shutdown of camera-related protocol handlers and exposes events for camera position and rotation changes.

Bertec.CameraContainerCallbacks.OnResetCameraRotation

Event triggered to reset the camera rotation to its default orientation.

Bertec.CameraContainerCallbacks.OnSetCameraRotation

Event triggered to set the camera rotation to a specific axis and angle.

newRotationAngle : the axis to set and the new rotation for it.

Bertec.CameraContainerCallbacks.OnResetCameraPosition

Event triggered to reset the camera position to its default value for a given axis.

resetAxis : the axis to reset

Bertec.CameraContainerCallbacks.OnSetCameraPosition

Event triggered to set the camera position to a specific value for a given axis.

resetAxis : the axis to set and the new position for it.

Bertec.CameraContainerCallbacks.Init()

Initializes the camera container callbacks and registers protocol command handlers. Called automatically; do not directly call

Bertec.CameraContainerCallbacks.Shutdown()

Shuts down the camera container callbacks and unregisters protocol command handlers. Called automatically; do not directly call

---

Bertec.ShowCognitiveData

Data for showing a cognitive display, including the choice ID and optional suboption.

Bertec.ShowCognitiveData.ID

The CognitiveChoice to show.

Bertec.ShowCognitiveData.SubOption

The suboption, if any.

Bertec.CognitiveDisplayedData

Data representing a displayed cognitive challenge, including challenge, response, text, color, and timestamp.

Bertec.CognitiveDisplayedData.Challenge

The question that the UI should display on the screen for the user to ask the subject.

Bertec.CognitiveDisplayedData.Response

The answer to the question that would be correct. The UI should record this and display.

Bertec.CognitiveDisplayedData.Text

The text that was shown; may not be the same as Challenge. UI should record this.

Bertec.CognitiveDisplayedData.Color

The color of the text, in RGB colors. The UI should record this.

Bertec.CognitiveDisplayedData.AtTime

The time at which the cognitive display was shown.

Bertec.CognitiveHiddenData

Data representing when a cognitive display is hidden, including the timestamp.

Bertec.CognitiveHiddenData.AtTime

The time at which the cognitive display was hidden.

Bertec.CognitiveEvents

Events and methods for handling cognitive display RPC commands and notifications.

Bertec.CognitiveEvents.ShowCognitiveDisplay

Event triggered when a cognitive display should be shown.

showWhat : the type of cognitive display to show

Bertec.CognitiveEvents.HideCognitiveDisplay

Event triggered when a cognitive display should be hidden.

void CognitiveDisplayed(string , string)

Notifies the server that a cognitive challenge has been displayed, with question and answer.

question : The challenge/question shown to the user.
answer : The correct answer to the challenge.

void CognitiveDisplayed(string , string , string , string)

Notifies the server that a cognitive challenge has been displayed, with full display details.

text : The text shown to the user.
color : The color of the text (RGB).
challenge : The challenge/question shown to the user.
response : The correct answer to the challenge.

Bertec.CognitiveEvents.CognitiveHidden()

Notifies the server that the cognitive display has been hidden.

Bertec.ExecuteOnMainThread

This is a static class that provides a handy functionality to resolve the gap with processing functions from one thread into the other. By using ExecuteOnMainThread.AddAction with the function in question and having your main loop code call ExecuteOnMainThread.ProcessActions it will "stack" the function calls until the main thread dequeues them. This is useful when processing incoming RPC comments that need to use a function that would manipulate a GUI or 3d object.

void AddAction(Action)

Adds the passed Action object to the queue to execute later on the main thread. The action will be executed in the order it was added in the main unity GUI/3D thread.

act : The action to perform. Ex: () => { CommandAddObject(connectionID, keyValuePairs); }

Bertec.ExecuteOnMainThread.ClearActions()

Removes all pending queue actions from the main thread execute queue.

Bertec.KeyPointVisualizerEvents

Provides events and data structures for handling keypoint visualizer changes and updates in the scene.

Bertec.KeyPointVisualizerEvents.VisualizerMode

The mode that the scene is expected to be in. This depends on both the Bertec.SceneFeatures.Keypoints setting and the UI selection.

Bertec.KeyPointVisualizerEvents.VisualizerMode.None

No visualizer mode selected. Not commonly triggered or handled.

Bertec.KeyPointVisualizerEvents.VisualizerMode.FirstPerson

Only available if Bertec.SceneFeatures.KeypointOptions.KeypointsAllowed is set. Mode when the user selects First Person. You scene will get KeypointUpdated with new values and ids

Bertec.KeyPointVisualizerEvents.VisualizerMode.ThirdPerson

Only available if Bertec.SceneFeatures.KeypointOptions.KeypointsAllowed is set. Mode when the user selects Third Person You scene will get KeypointUpdated with new values and ids

Bertec.KeyPointVisualizerEvents.VisualizerMode.CoP

Only available if Bertec.SceneFeatures.KeypointOptions.CoPAllowed is set. You scene will get CoPVisualizerUpdated with the selected visualizer object, if any

Bertec.KeyPointVisualizerEvents.FORCEPLATEKEYPOINTTAG

Key used to identify forceplate keypoints. See VisualizerChanged and KeypointPositionUpdated

---

Bertec.KeyPointVisualizerEvents.VisualizerModeData

Data structure for visualizer mode changes.

Bertec.KeyPointVisualizerEvents.VisualizerModeData.Mode

The new visualizer mode.

Bertec.KeyPointVisualizerEvents.ChangeVisualizerData

Data structure for visualizer selection changes.

Bertec.KeyPointVisualizerEvents.ChangeVisualizerData.Key

The key identifying the visualizer.

Bertec.KeyPointVisualizerEvents.ChangeVisualizerData.Value

The value associated with the visualizer key.

Bertec.KeyPointVisualizerEvents.KeypointPosition

Represents a single keypoint's position in 3D space.

Bertec.KeyPointVisualizerEvents.KeypointPositionsUpdatedData

Data structure for updated keypoint positions.

Bertec.KeyPointVisualizerEvents.KeypointPositionsUpdatedData.Keypoints

Dictionary mapping keypoint IDs to their positions.

Bertec.KeyPointVisualizerEvents.VisualizerModeChanged

Invoked when the UI changes the Visualizer type, if any

mode : The new selected.

Bertec.KeyPointVisualizerEvents.VisualizerChanged

Invoked when the UI changes a visualizer selection. The new string will the be key name that the scene should handle. For the COP, you should compare the key against the static string.

key : The key identifying the visualizer.
value : The value associated with the visualizer key.

Bertec.KeyPointVisualizerEvents.KeypointPositionUpdated

Invoked when the keypoint visualizer data is updated. Your scene should move the keypoint id to the passed vector. For the COP, you should compare the key against the static string.

key : The key identifying the keypoint.
position : The new position of the keypoint as a .

Bertec.SingleMarkerTrackerPosition

The position and occlusion state of a single marker tracker.

Bertec.DualMarkerTrackerPositions

The positions of two marker trackers (left and right).

Bertec.ForceLineValues

The force line values for left and right.

Bertec.MarkerTrackerEvents

Events for marker tracker updates and force line values.

Bertec.MarkerTrackerEvents.SingleMarkerTrackerUpdated

Event triggered when a single marker tracker position is updated.

position : The updated marker tracker position.

Bertec.MarkerTrackerEvents.DualMarkerTrackersUpdated

Event triggered when both left and right marker tracker positions are updated.

positions : The updated dual marker tracker positions.

Bertec.MarkerTrackerEvents.HeadTrackerUpdated

Event triggered when the head marker tracker position is updated.

position : The updated head marker tracker position.

Bertec.MarkerTrackerEvents.ForceLinesUpdated

Event triggered when force line values are updated.

values : The updated force line values.

Bertec.ObstacleHitmissCount

Represents the hit and miss count data for obstacles.

Bertec.ObstacleShownHiddenData

When an obstacle is shown or hidden.

Bertec.ObstacleEvents

Provides events and methods for handling obstacle-related actions and communication between the scene and PC.

Bertec.ObstacleEvents.OnResetHitMissCounts

Invoked when the PC side wants the Scene to reset the hit/miss counts for the Obstacle. Typically triggered when starting/restarting a protocol

Bertec.ObstacleEvents.ObstacleDisplayed()

Call this when an Obstacle shows up on the screen; the PC side may elect to trigger an event with this

Bertec.ObstacleEvents.ObstacleHidden()

Call this when an Obstacle is removed from the screen; the PC side may elect to trigger an event with this

void UpdateHitMiss(int , int)

Call to update the running hit/miss count for Obstacles. The PC side will keep track of this and display info and recording

hit : The current hit count.
miss : The current miss count.

Bertec.ObstacleEvents.ResetHitMissCounts()

Triggers the reset of hit/miss counts for obstacles.

Bertec.OptionsChangedEventData

Event data for when protocol options have changed.

Bertec.OptionsChangedEventData.TestGUID

The protocol GUID that is requesting the change.

Bertec.OptionsChangedEventData.Options

The options to pass to the scene.

Bertec.PackageInfoInterface

Interface for retrieving package version information (platform specific)

void GetPackageVersion(string)

Gets the version string for a specific package.

packageName : The name of the package.

Bertec.PackageInfoInterface.GetPackageVersions()

Gets a dictionary of all package names and their version strings.

void GetPackageVersion(string)

Gets the version string for a specific package.

packageName : The name of the package.

Bertec.PackageInfo.GetPackageVersions()

Gets a dictionary of all package names and their version strings.

Bertec.PostProcessingEffectsMoniter

Monitors and manages post-processing effects, raising events when effects are changed.

Bertec.PostProcessingEffectsMoniter.PostProcessingEffectChanged

Event triggered when a post-processing effect is changed.

effectData : The effect and level to apply.

Bertec.PostProcessingLayerHelper

Provides access to post-processing layer helper functionality.

void EnablePostProcessLayer(UnityEngine.Camera , bool)

Enables or disables the post-processing layer on the specified camera.

mainCamera : The camera to modify.
enabled : True to enable, false to disable.

void AddPostProcessLayer(UnityEngine.Camera , UnityEngine.GameObject , UnityEngine.Transform)

Adds a post-processing layer to the specified camera.

mainCamera : The camera to add the layer to.
cameraObject : The GameObject associated with the camera.
cameraTransform : The transform of the camera.

Bertec.FrameworkInitType

The initialization stages for the Bertec protocol framework.

Bertec.FrameworkInitType.PrebuildExec

Called during the build step (may be called multiple times).

Bertec.FrameworkInitType.RegisterObjectStructs

Called during SubsystemRegistration to register all object structs

Bertec.FrameworkInitType.BeforeSubsystem

Called during SubsystemRegistration BEFORE the framework core bits.

Bertec.FrameworkInitType.AfterSubsystem

Called during SubsystemRegistration AFTER the framework core bits.

Bertec.FrameworkInitType.BeforeAssemblies

Called during AfterAssembliesLoaded BEFORE the framework core bits.

Bertec.FrameworkInitType.AfterAssemblies

Called during AfterAssembliesLoaded AFTER the framework core bits.

Bertec.FrameworkInitType.BeforeSplashScreen

Called during BeforeSplashScreen BEFORE the framework core bits.

Bertec.FrameworkInitType.AfterSplashScreen

Called during BeforeSplashScreen AFTER the framework core bits.

Bertec.FrameworkInitType.BeforeSceneLoad

Typically not used; called before scene load. You would prefer Start or Awake more than anything

Bertec.FrameworkInitType.AfterSceneLoad

Called after scene load.

Bertec.FrameworkInitAttribute

This attribute extends the Unity.RuntimeInitializeOnLoadMethod attribute to allow more finer-grained control from within the Bertec Framework. The init levels allow both before and after each stage, so you can be sure to have things initialized in more or less the proper order.

Bertec.ProtocolFramework

Provides initialization and shutdown logic for the Bertec protocol framework, including client/server/bridge startup and event hooks.

Bertec.ProtocolFramework.OnTerminate

Event triggered when the framework is terminating.

Bertec.ProtocolFramework.OnTerminating

Event triggered when the framework is terminating, with a flag indicating if it is a hard/fast shutdown.

Bertec.ProtocolFramework.IsServer

Gets whether the framework is running in server mode.

Bertec.ProtocolFramework.IsBridge

Gets whether the framework is running in bridge server mode.

Bertec.ProtocolFramework.IsClient

Gets whether the framework is running in client mode (most scenes will have this)

Bertec.ProtocolFramework.Terminating

Gets whether the framework is currently terminating.

Bertec.ProtocolFramework.StartType

The type of service the framework is running as.

void Terminate(bool)

Call all the shutdown methods and disconnect the RPC, then exit the application. This will do a hard-kill of the running app, not just send it to the back and suspend it. Use instead of UnityEngine.Application.Quit()

Note: Note that the Framework does this for you automatically and you should never do this unless you have a VERY specific use-case.

hardAndFast : If true, does not attempt to gracefully shut down first; app exit may leave resources left open.

Bertec.StartupTypeAttribute

This attribute is used by the Framework to flag that the Unity Project is classic or server scene; by default, it will be "client" scene so the Kinamoto Customer scenes don't need to do any extra work.

Bertec.PXRServiceBridgeInitAttribute

This attribute must be on the PXRServiceBridge.Init function otherwise things won't work right on the headset What this really does is basically tie this in the FrameworkInitType.BeforeSubsystem so it gets called at the right time

---

Bertec.PlateInfo

Represents information about a force plate, including type, dimensions, firmware, and serial.

Bertec.PlateInfo.PlateType

The type of force plate.

Bertec.SceneMovement

Specifies the type of scene movement.

Bertec.SceneMovement.Static

Scene doesn't move.

Bertec.SceneMovement.SwayReferenced

Scene moves with the SwayAngle value (nodding rotation).

Bertec.SceneMovement.PlateReferenced

Scene moves with the translation position and rotation (camera in/out+rotation).

Bertec.SceneMovement.RollRotation

Scene rolls left/right (shoulder, random l/r direction, 15 degrees @ 45 deg/sec)).

Bertec.ProtocolOptionChangedEventHandler

Handles protocol option changes and exposes events for option updates.

Bertec.ProtocolOptionChangedEventHandler.Scene

The current scene name.

Bertec.ProtocolOptionChangedEventHandler.SceneChanged

Event triggered when the scene changes.

sceneName : name of the scene
testGuid : protocol guid

Bertec.ProtocolOptionChangedEventHandler.SubjectHeight

The subject's height in millimeters.

Bertec.ProtocolOptionChangedEventHandler.SubjectHeightChanged

Event triggered when the subject height changes.

Bertec.ProtocolOptionChangedEventHandler.AudioFeedback

Indicates if audio feedback is enabled.

Bertec.ProtocolOptionChangedEventHandler.AudioFeedbackChanged

Event triggered when audio feedback changes.

Bertec.ProtocolOptionChangedEventHandler.OnOptionChanged

Event triggered whenever an unhandled option is changed.

Bertec.ProtocolOptionChangedEventHandler.ClearAllEventHandlers()

Clears all event handlers by resetting them to empty delegates.

Bertec.ProtocolOptions

This class connects to the protocol-command web socket interface and allows you to issue commands over that. Some protocols support the ProtocolCommand interface, allowing for commands to be sent to them. These commands can be simple things (ex: SetScore) with no response or more complex functions that return a single future result (ex: GetScoreForHitCount). Other commands return *periodic* updates; these periodic updates can be considered either event signals (ex: some condition has changed) or continual data updates (ex: live calculations against the data flow)

Note: This is basically a convenience wrapper around the ProtocolRPC functions.

Bertec.ProtocolOptions.OnOptionChanged

Event triggered when an option changes. The string is the key, the object is the new value.

key : The key for the option. See the SceneOptions.
value : The value for the changed option.

Bertec.ProtocolOptions.AllOptionEventsExecuteOnMainThread

If true, all option events automatically execute on the main thread via Bertec.ExecuteOnMainThread.AddAction.

Bertec.ProtocolOptions.ResetOptions()

Resets the current options to the initial options parsed from the command line or launch intent.

void IsSet(string)

Determines whether the specified key is set in the current options.

key : The option key.

void UpdateOptions(Collections.Generic.Dictionary<String , Object>)

Updates multiple options from the provided dictionary. Triggers multiple OnOptionChanged events.

dict : Dictionary of options (key:value pairs) to update.

void UpdateOption(string , object)

Updates a single option and triggers the OnOptionChanged event.

key : The option key.
value : The new value.

Bertec.ProtocolOptions.CurrentOptions

Gets the current protocol options as a dictionary.

void DictionaryValue(string)

Gets the value of the specified key as a dictionary.

key : The option key.

void ArrayValue(string)

Gets the value of the specified key as a string array.

key : The option key.

void StringValue(string)

Get the string value for the given option; if it doesn't exist, an empty string is returned.

key : The option key

void StringValue(string , string)

Get the string value for the given option; if it doesn't exist, the default val string is returned.

key : The option key
val : Default value if key does not exist

void IntValue(string , int)

Get the integer value for the given option; if it doesn't exist, the default val is returned.

key : The option key
val : Default value if key does not exist

void Unsigned64Value(string , ulong)

Get the unsigned 64 bit integer value for the given option; if it doesn't exist, the default val is returned.

key : The option key
val : Default value if key does not exist

void FloatValue(string , float)

Get the float value for the given option; if it doesn't exist, the default val is returned.

key : The option key
val : Default value if key does not exist

void BoolValue(string , bool)

Get the boolean value for the given option; if it doesn't exist, the default val is returned. This function handles both numbers (0,1,-1) and strings (false, true). 0 is treated as false, anything else is treated as true.

key : The option key
val : Default boolean value if key does not exist

void ColorValue(string , UnityEngine.Color)

Get the given option as a Unity Color object. This method handles both discrete numbers (1,2,3) and HTML-style hex colors (#010203). Both RGB and RGBA variants are supported.

key : The option key
val : Default color value if key does not exist

void Vector2Value(string , UnityEngine.Vector2)

Get the given option as a Unity Vector2 object, that is, two discrete numbers (1.0,2.5)

key : The option key
val : Default Vector2 value if key does not exist

void Vector3Value(string , UnityEngine.Vector3)

Get the given option as a Unity Vector3 object, that is, three discrete numbers (1.0,2.5,3.9)

key : The option key
val : Default Vector3 value if key does not exist

Bertec.ClientConnectedDisconnected

Event data for when a client connects or disconnects.

Bertec.ProtocolRPC

Class that manages the network connection and the rpc commands. Does the command dispatch (callbacks) for the registered handles and etc.

Bertec.ProtocolRPC.ConnectedIP

The connected IP address if connected, otherwise an empty string.

Bertec.ProtocolRPC.ConnectedURL

The connected URL if connected, otherwise an empty string.

Bertec.ProtocolRPC.IsConnected

Returns TRUE if the web socket interface is currently connected. This does not mean that communication is actually taking place but just that a server of some type exists on the other end.

Bertec.ProtocolRPC.OnConnected

Event triggered when the RPC system is connected to the PC

Bertec.ProtocolRPC.OnDisconnected

Event triggered when the RPC system is disconnected from the PC

void IssueResponse(ulong , object)

Sends the response from the handled command object and parms over in a fire-and-forget manner; no callbacks are set up to be handled and there is no return value.

Note: IssueResponse is the same as IssueCommand and calls that if responseID is non zero

responseID : The response uuid that will be sent back; the other side will handle this.
parms : An object (typically an anonymous type) of parms to be sent. These will be converted to a JSON object string.

void IssueResponse(ulong , byte[])

Sends the response from the handled command object and parms as a byte array in a fire-and-forget manner.

Note: IssueResponse is the same as IssueCommand and calls that if responseID is non zero

responseID : The response uuid that will be sent back.
parms : The byte array of parameters to send.

void IssueResponse(ulong)

Sends the response from the handled command object with no parameters in a fire-and-forget manner.

Note: IssueResponse is the same as IssueCommand and calls that if responseID is non zero

responseID : The response uuid that will be sent back.

void IssueResponse<T>(ulong , T)

Sends the response from the handled command object and strongly-typed parameters in a fire-and-forget manner.

Note: IssueResponse is the same as IssueCommand and calls that if responseID is non zero

responseID : The response uuid that will be sent back.
parms : The parameters to send.

void IssueCommand(ulong , object)

Sends the command and the parms over in a fire-and-forget manner; no callbacks are set up to be handled and there is no return value.

cmdID : The command to be sent to the web socket interface; these are currently defined on the server.

parms : An object (typically an anonymous type) of parms to be sent. These will be converted to a JSON object string.

void IssueCommand(Bertec.RPCCommands.Cmd , object)

Sends the command and the parms over in a fire-and-forget manner using a command enum.

cmdID : The command enum to be sent to the web socket interface.
parms : An object of parms to be sent.

void IssueCommand(Bertec.RPCCommands.Cmd , byte[])

Sends the command and the parms as a byte array over in a fire-and-forget manner.

cmdID : The command to be sent to the web socket interface.
parms : The byte array of parameters to send as-is.

void IssueCommand<T>(ulong , T)

Sends the command and strongly-typed parameters over in a fire-and-forget manner using a command enum.

cmdID : The command enum to be sent to the web socket interface.
parms : The byte array of parameters to send.

void IssueCommand<T>(Bertec.RPCCommands.Cmd , T)

Sends the command and strongly-typed parameters over in a fire-and-forget manner.

cmdID : The command to be sent to the web socket interface.
parms : The parameters to send.

void IssueCommand(ulong)

Sends the command with no parameters over in a fire-and-forget manner.

cmdID : The command to be sent to the web socket interface.

void IssueCommand(ulong , object , Action<Bertec.RPCCommandPacket>)

Sends the command and the parms with a callback id that will invoke the completionCallback method if and when the host returns a value. This is expected to result in a single updates which will call the completionCallback once and then remove the uuid and callback from the internal table.

cmdID : The command to be sent to the web socket interface; these are currently defined on the server.
parms : An object (typically an anonymous type) of parms to be sent. These will be converted to a JSON object string.
completionCallback : The method that should be invoked when the server returns back a result. The method will be called with the raw JSON string and an generic object parsed from the JSON string.

void IssueProtocolCommand(string , object)

Issues a protocol command under the PROTOCOLCOMMAND command id with parms and no completion callback. The main desktop protocol handler will respond to whatever this is. This is specific to the protocol itself and may not be compatible between different protocols

command : the protocol-specific command to issue. Sent as-is
parms : anonymous collection parms for the command

void AddCommandHandler(ulong , Action<Bertec.RPCCommandPacket>)

Adds a callback handler for the given command command id.

cmdID : The command id that should be matched.
completionCallback : The method that should be invoked when the handler recieves the command id. The method will be called with the RPCCommandPacket which can be used to unpack the data and optionally send back a response.

void AddCommandHandler(Bertec.RPCCommands.Cmd , Action<Bertec.RPCCommandPacket>)

Adds a callback handler for the given command enum id.

cmdID : The command enum id that should be matched.
completionCallback : The method that should be invoked when the handler receives the command id.

void RemoveCommandHandler(ulong)

Removes the callback for the given command id. This does *NOT* tell the server to stop sending results for the command; it just stops responding to that command. If you need the server to stop sending data, you should send whatever matching "stop" command is available for the previous command, if applicable.

cmdID : The command id that was passed into AddCommandHandler

void RemoveCommandHandler(Bertec.RPCCommands.Cmd)

Removes the callback for the given command enum id.

cmdID : The command enum id that was passed into AddCommandHandler

---

Bertec.SceneInfoAttribute

Attribute tag class that is used by the build step to generate a json file of all the scene info that Kinamoto can then read

Bertec.SceneInfoAttribute.Name

The display name of the scene.

Bertec.SceneInfoAttribute.Key

The unique id key for the scene.

Bertec.SceneInfoAttribute.Scene

The scene path.

Bertec.SceneInfoAttribute.Description

The description of the scene to show in the ui.

Bertec.SceneInfoAttribute.AKA

Alternate keys or names for the scene.

Bertec.SceneChoiceItem.Key

Required. The unique id within this option that will be used to track the selected option choice (can be same in different options, does not need to be globally unique). The Key is used to track the value, so it must be unique within the option. The Name is what is displayed to the user. Typically, the choice's 'value' is the same as the key - so for example, if you have a Key equal to '8', the value would be 8. Key can be a string like 'blue' or a number like '2.5'. The scene handler is expected to understand what to do with it.

Bertec.SceneChoiceItem.Name

Optional. The display name that is shown on the UI display.

Bertec.SceneChoiceItem.Description

Optional. Describes what the event is (can be used for the UI or other documentation).

Bertec.SceneChoiceItem()

Initializes a new instance of the class.

Bertec.SceneChoiceItem(System.Object)()

Initializes a new instance of the class.

v : The value to be used for the Key and Name.

Bertec.SceneChoiceItem(System.Object, System.Object)()

Initializes a new instance of the class.

k : The value to be used for the Key.
n : The value to be used for the Name.

Bertec.SceneChoices

A collection of scene choice items and a default choice.

Bertec.SceneChoices.Items

Array of choices that will be displayed for the user to see. Choices will be shown in the same order as they appear in the array. Not used for the Title type.

Bertec.SceneChoices.DefaultChoice

Optional. If the item's value has not been set by other means (such as reloading a previous test), then the default value will be set using this from this. Depending on the item, this can be a value from the list of choices (type = panel), the word checked or unchecked (type = checkbox), a string (type = edit) or a date string (type = date). Not used for the Title type.

Bertec.SceneChoices(Bertec.SceneChoiceItem)()

Initializes a new instance of the class with a single item.

oneitem : The single choice item.

Bertec.SceneChoices(System.Collections.Generic.List{Bertec.SceneChoiceItem}, System.String)()

Initializes a new instance of the class with a list of choices and a default.

choices : The list of choices.
defaultChoice : The default choice value.

Bertec.CognitiveChoiceItem

A cognitive choice item, which may have sub-options.

Bertec.CognitiveChoiceItem.SubOptions

Optional. Possible sub options that can be selected for the Cognitive choice. If not set/defined/only one, then the UI will not show a sub-option panel.

Bertec.CognitiveChoiceItem.SubOptionName

Required if SubOptions is defined. What is shown in the UI as the label lead-in for the SubOptions

---

Bertec.CognitiveChoices

A collection of cognitive choice items and a default choice.

Bertec.CognitiveChoices.Items

Array of Cognitive choices that will be displayed for the user to see. Items will be shown in the same order as they appear in the array.

Bertec.CognitiveChoices.DefaultChoice

The default choice value.

Bertec.CognitiveChoices(Bertec.CognitiveChoiceItem)()

Initializes a new instance of the class with a single item.

oneitem : The single cognitive choice item.

Bertec.CognitiveChoices(System.Collections.Generic.List{Bertec.CognitiveChoiceItem}, System.String)()

Initializes a new instance of the class with a list of choices and a default.

choices : The list of cognitive choices.
defaultChoice : The default choice value.

Bertec.CameraRotations

The available camera rotation modes.

Bertec.SceneFeatures

SceneFeatures are separate from the Scene Options and are used to drive more specific effects that have broad applicable terms, and more rich UI interfaces than standard options. Some of the capabilities are handled fully by the Framework (ex: PostProcessing, Rotation), but others will require your scene handle event triggers and values.

Bertec.SceneFeatures.VisualFlow

Implies the scene moves with the treadmill. The UI will provide a belt sync checkbox, fixed scene velocity value, and belt speed multiplier.

Bertec.SceneFeatures.HasPostProcessing

Implies Vignette, Grain, and possibility other effects in the UI. The framework will handle these features.

Bertec.SceneFeatures.HasObstacles

Implies the protocol will either trigger or respond to obstacles in the scene, and expects to be notified if hit/miss. How the obstacles are presented and controlled is up to the scene, including any options to turn them on/off.Turning this on means you intend to use the ObstacleEvents class.

Bertec.SceneFeatures.HasCameraPosition

Implies position x/y/z inputs and controls in the UI. The framework will handle the positioning.

Bertec.SceneFeatures.CameraRotation

Implies rotation x/y/z inputs and controls in the UI. The framework will handle the rotation.

Bertec.SceneFeatures.CoPKeypointVisualizers

If CoPVisualizer or KeypointVisualizer set, has a special Key Point Visualizer panel in the UI that fills in from the rest of the options. These are tied into the Bertec.KeyPointVisualizerEvents class and the Bertec.KeyPointVisualizerEvents.VisualizerMode enum

Bertec.SceneFeatures.KeypointVisualizers

If set, then the Key Point Visualizer UI will have the Key Points option available, and your code should respond to various keypoint actions. Ideally, there should be at least one and preferably three or more items to choose from. For 1st Person, there should be two or more, and for 3rd person, there should be three. If there is less than needed 2 (1stperson) or 3 (3rdperson), then this will limit how many Keypoints there can be selected in the UI. This also implies that the UI has both 1st and 3rd person options, which further controls that is enabled

Bertec.SceneFeatures.Cognitive

If set, has a special panel of Cognitive Choices with sub-options/difficulties

void AddKeypointVisualizer(string , string)

Adds a keypoint visualizer option.

void AddKeypointVisualizer(string , string , Bertec.SceneChoices)

Adds a keypoint visualizer option with choices.

void AddKeypointVisualizers(Collections.Generic.List<Bertec.KeyPointOption>)

Adds multiple keypoint visualizer options.

Bertec.SceneProperty

SceneProperty items are list of things that the UI can set in the Scene.

Bertec.SceneEvent

SceneEvent items are list of things that can be triggered in either the UI or the Scene

Bertec.SceneEvent.IsTriggerableFromUI

If true, then the UI should display a button to trigger some event on the Unity side or otherwise connect it.

Bertec.SceneEvent.IsTriggerableFromScene

If true, then the scene will trigger the ui with this event. The UI should allow the scene to control some action.

Bertec.SceneOption

A scene option, including type, requirement, key, name, description, grouping, and choices.

Bertec.SceneOption.OptionType

The type of option.

Bertec.SceneOption.OptionType.None

No option.

Bertec.SceneOption.OptionType.Checkbox

Simple checkbox. Choices will be ignored.

Bertec.SceneOption.OptionType.Choicelist

Could be a combo box or a set of radio buttons. Choices will be used to present the items.

Bertec.SceneOption.OptionType.Range

Typically a slider. Choices are used to build the range.

Bertec.SceneOption.OptionType.Text

Any value, UI shows an edit box. Choices will be ignored.

Bertec.SceneOption.RequirementType

RequirementType is used to tell the UI if the option needs to be set or not. It is up to the UI to handle this.

Bertec.SceneOption.RequirementType.NotRequired

Value can be blank (default type, no requirement).

Bertec.SceneOption.RequirementType.Required

Value cannot be blank, but can be default.

Bertec.SceneOption.RequirementType.RequiredAlways

Value must be changed, even if the default is used (user must touch).

Bertec.SceneOption.Decimals

When calling SetRange with float values, use this to determine how many decimal points to show

Bertec.SceneOption.Type

Used to determine which UI element to display.

Bertec.SceneOption.Requirement

Optional. If set, then this option must have a valid value before the test can be started. Not used for the Title type.

Bertec.SceneOption.Key

Required. The unique id that will be used to track the option and save/restore from database. Also used to send/return from the unity scene and pc UI. Can be the same value in different scenes, but must be unique within a scene.

Bertec.SceneOption.Name

Optional. The display name that is shown on the UI display on the button/callout and the title box if any.

Bertec.SceneOption.Description

Optional. Describes what the option is (can be used for the UI or other documentation).

Bertec.SceneOption.Grouping

Optional; defines the grouping section the option belongs to (used by UI to organize things).

Bertec.SceneOption.Group

Easy alias to allow setting a single top-level group.

Bertec.SceneOption.Subgroup

Easy alias to allow setting a the 2nd level sub group once the top level has been set.

void SetGroup(string[])

Easy function to set the group with a list of strings

Bertec.SceneOption.Choices

Valid for Type.Choicelist and Type.Range only. Array of choices that will be displayed for the user to see. Choices will be shown in the same order as they appear in the array. Not used for the Title type.

Bertec.SceneOption(Bertec.SceneOption.OptionType, System.String, System.String, System.String)()

Initializes a new instance of the class with type, key, name, and description.

void AddChoice(Bertec.SceneChoiceItem , bool)

Adds a choice to the option.

void AddChoice(object)

Adds a choice to the option.

void AddChoice(object , object , bool)

Adds a choice to the option with key and name.

void SetRange(float , float , float , float , Bertec.SceneOption.Decimals)

Sets a float range for the option.

void SetRange(int , int , int , int)

Sets an integer range for the option.

void SetDefault(Bertec.SceneChoiceItem)

Sets the default choice.

void SetDefault(object)

Sets the default choice.

void SetDefault(bool)

Sets the default choice.

Bertec.KeyPointOption

A keypoint visualizer option.

Bertec.KeyPointOption.Key

Required. The unique id that will be used to track the option and save/restore from database. Also used to send/return from the unity scene and pc UI. Can be the same value in different scenes, but must be unique within a scene.

Bertec.KeyPointOption.Name

Optional. The display name that is shown on the UI display on the button/callout and the title box if any.

Bertec.KeyPointOption.Description

Optional. Describes what the option is (can be used for the UI or other documentation).

Bertec.KeyPointOption.Choices

Array of choices that will be displayed for the user to see. Choices will be shown in the same order as they appear in the array.

Bertec.KeyPointOption(System.String, System.String)()

Initializes a new instance of the class with key and name.

Bertec.KeyPointOption(System.String, System.String, Bertec.SceneChoices)()

Initializes a new instance of the class with key, name, and choices.

void AddChoice(Bertec.SceneChoiceItem , bool)

Adds a choice to the keypoint option.

void AddChoice(object)

Adds a choice to the keypoint option.

void AddChoice(object , object , bool)

Adds a choice to the keypoint option with key and name.

void SetDefault(Bertec.SceneChoiceItem)

Sets the default choice.

void SetDefault(object)

Sets the default choice.

void SetDefault(bool)

Sets the default choice.

Bertec.SceneInfo

Represents information about a scene, including key, name, description, icon, features, options, events, and properties.

Bertec.SceneInfo.ScenePath

Internal value that is used by the SceneManager to locate the scene to load. Not exposed to the UI.

Note: The SceneListManager will populate this field based on the primary Key value. If your scene would prefer to set this itself,
do so in the constructor.

Bertec.SceneInfo.Key

Required. The unique id is used to find and display the scene (will be tied to an actual scene asset). Typically this will be just the filename part of the scene path - ex: Foobar for "Assets/folder/Foobar.unity"

Note: If more than one possible key value can be used to reference the scene, override the property and return those as well.

Bertec.SceneInfo.Name

Required. The name that is displayed in the UI.

Bertec.SceneInfo.Description

Optional; describes what the scene is (can be used for the UI).

Bertec.SceneInfo.Icon

Optional; a byte array of the icon image to display in the UI (typically png format). If the scene does not provide this directly, the SceneListManager will attempt to find one in the Streaming Assets folder

Bertec.SceneInfo.Features

Optional; not used for most scenes but the UI can use to determine specific features.

Bertec.SceneInfo.Options

All the options and choices. Technically optional, but you'll want to have some.

Bertec.SceneInfo.Events

List of Events the scene and send or receive. The scene may trigger these and send to the UI, and the UI may also trigger these and send back

Bertec.SceneInfo.Properties

List of Properties the scene can understand (or send to the UI). Similar to options without any choices (may be removed later)

Bertec.SceneInfo()

The default constructor will take the Key, Name, and other items from the attribute tag, if set

Bertec.SceneInfo(System.String, System.String)()

Initializes a new instance of the class. This is desrired constructor to use.

key : The value to be used for the Key. This will be used to generate the ScenePath unless that is set explictly.
name : The value to be used for the Name.

Bertec.SceneInfo(System.String, System.String, System.String)()

Initializes a new instance of the class with key, name, and scene file.

key : The value to be used for the Key. This will be used to generate the ScenePath unless that is set explictly.
name : The value to be used for the Name.
scenefile : The path to the local project scene

void SetIconData(string)

Sets the icon data from a file.

imageName : The image file name.

void AddOption(Bertec.SceneOption)

Adds an option to the scene.

void AddOption(Bertec.SceneOption.OptionType , string , string , string)

Adds an option to the scene.

void AddCheckbox(string , string , string)

Adds a checkbox option (SceneOption.OptionType.Checkbox) to the scene.

void AddList(string , string , Collections.Generic.List<Bertec.SceneChoiceItem> , string)

Adds a list option to the scene.

void AddList(string , string , string , Collections.Generic.List<Bertec.SceneChoiceItem> , string)

Adds a list option to the scene with description.

void AddList(string , string , string)

Adds a list option (SceneOption.OptionType.Choicelist) to the scene.

void AddRange(string , string , string , float , float , float , float , Bertec.SceneOption.Decimals)

Adds a float range option to the scene.

void AddRange(string , string , float , float , float , float , Bertec.SceneOption.Decimals)

Adds a float range option to the scene.

void AddRange(string , string , string , int , int , int , int)

Adds an integer range option to the scene with description.

void AddRange(string , string , int , int , int , int)

Adds an integer range option to the scene.

Bertec.SceneInfo.AKA

Returns a list of all the keys that can be used to identify this scene. By default this will just be the and values, but can be more.

Note: Typically used when a possible "Scene" option references a historical value that has changed or can be mapped to another scene.

Bertec.SceneInfo.IconPath

Tries to find the matching icon from the given key name. Used during the init/build phase to generate the optional icon

Bertec.SceneInfo.IsValid

Gets whether the scene info is valid (has a key and name).

Bertec.SceneInfoList

Represents a list of scene info objects and related metadata.

Bertec.SceneInfoList.PackageName

The package name;set from Application.identifier

Bertec.SceneInfoList.PackageVersion

The package version. Used to figure out if the installed package needs updated against the scene info.

Bertec.SceneInfoList.ProductName

The product name; set from Application.productName

Bertec.SceneInfoList.FrameworkVersion

The framework version.

Bertec.SceneInfoList.FrameworkProtocol

The framework protocol version.

Bertec.SceneInfoList.Scenes

The list of scenes.

Bertec.ChangeSceneData

Data for changing scenes, including force reload, scene name, test GUID, and options.

Bertec.ChangeSceneData.ForceReload

If true, forces the scene to reload.

Bertec.ChangeSceneData.Scene

The scene to change to.

Bertec.ChangeSceneData.TestGUID

The protocol GUID that is requesting the change.

Bertec.ChangeSceneData.Options

Options to pass to the scene.

Bertec.ChangeSceneResult

Result of a scene change operation.

Bertec.ChangeSceneResult.Success

True if the scene change was successful.

Bertec.ChangeSceneResult.PackageName

The package name.

Bertec.ChangeSceneResult.Scene

The scene that was changed to.

Bertec.CurrentSceneData

Data for the current scene, including package name, scene, and test GUID.

Bertec.CurrentSceneData.PackageName

The package name.

Bertec.CurrentSceneData.Scene

The current scene.

Bertec.CurrentSceneData.TestGUID

The test GUID.

void GetSceneInfoList(bool)

Retrieves a list of SceneInfo objects representing the scenes in the build settings. The build process uses this to generate the scene info json file, and the runtime manager uses this to get a list of scenes that can be loaded by key name or alias.

resolvePaths : A boolean value indicating whether to resolve the scene paths. False for the build, true at runtime.

Bertec.SystemAudioDeviceManagerInterface

Interface for system audio device management, providing methods to get and set audio volume.

void GetAudioVolumeInfo(out int , out int , out int)

Gets the minimum, maximum, and current audio volume levels.

_***min*** : The minimum volume level.
_***max*** : The maximum volume level.
_***level*** : The current volume level.

void SetAudioVolumeLevel(int)

Sets the audio volume to the specified level.

_***newlevel*** : The new volume level to set.

Bertec.SystemAudioDeviceManager

Methods and events for managing system audio device volume.

Bertec.SystemAudioDeviceManager.AudioVolumeChanged
Event triggered when the audio volume changes.

level : The new audio volume level.

Bertec.SystemDisplayDeviceManagerInterface

Interface for system display device management, providing methods for passthrough, screen, and brightness control.

Bertec.SystemDisplayDeviceManagerInterface.GetCurrentFPS()

Gets the current frames per second.

Bertec.SystemDisplayDeviceManagerInterface.ResetHeadsetPosition()

Resets the headset position. Does nothing on Windows

Bertec.SystemDisplayDeviceManagerInterface.TurnOffPassthrough()

Turns off passthrough mode.

Bertec.SystemDisplayDeviceManagerInterface.GetPassthroughTrackingState()

Gets the current passthrough tracking state.

void PassthroughChanged(bool)

Notifies that passthrough state has changed.

nextPassthroughState : The new passthrough state.

void EnableSeeThroughManual(bool)

Enables or disables see-through manually.

f : True to enable, false to disable.

void GetScreenOnOff(out bool)

Gets whether the screen is currently on.

screenCurrentlyOn : True if the screen is on.

void SetScreenOnOff(bool)

Sets the screen on or off.

screenOn : True to turn on, false to turn off.

void GetScreenBrightness(out int , out int , out int)

Gets the screen brightness range and current level.

_***min*** : Minimum brightness.
_***max*** : Maximum brightness.
_***level*** : Current brightness level.

void SetScreenBrightness(int)

Sets the screen brightness.

newlevel : The new brightness level.

Bertec.SystemDisplayDeviceManager

Methods and events for managing system display device state, including passthrough, screen, and brightness.

Bertec.SystemDisplayDeviceManager.PassThroughEnabled

Gets or sets whether passthrough is enabled.

Bertec.SystemDisplayDeviceManager.IsPassthrough

Gets or sets whether passthrough is currently active.

Bertec.SystemDisplayDeviceManager.ScreenCurrentlyOn

Gets or sets whether the screen is currently on.

Bertec.SystemDisplayDeviceManager.OnPassThroughEnabled
Event triggered when passthrough is enabled or disabled.

enabled : True if enabled, false if disabled.

Bertec.SystemDisplayDeviceManager.OnPassthroughChanged
Event triggered when passthrough state changes.

enabled : True if enabled, false if disabled.

Bertec.SystemDisplayDeviceManager.OnEnableHeadsetPassthrough

Event triggered to enable headset passthrough.

enabled : True to enable, false to disable.

Bertec.SystemDisplayDeviceManager.OnResetHeadsetPosition

Event triggered to reset headset position.

Bertec.SystemDisplayDeviceManager.BrightnessLevelChanged

Event triggered when brightness level changes.

level : The new brightness level.

Bertec.SystemDisplayDeviceManager.GetCurrentFPS()

Gets the current frames per second.

Bertec.SystemDisplayDeviceManager.ResetHeadsetPosition()

Resets the headset position (ignored on Windows).

Bertec.SystemDisplayDeviceManager.TurnOffPassthrough()

Turns off passthrough.

Bertec.SystemDisplayDeviceManager.GetPassthroughTrackingState()

Gets the current passthrough tracking state

void EnableHeadsetPassthrough(bool)

Sets the current passthrough state and triggers the OnEnableHeadsetPassthrough event.

f : True to enable, false to disable.

void EnableSeeThroughManual(bool)

Enables or disables see-through manually

f : True to enable, false to disable.

Bertec.Utils

Utility class providing helper methods for type conversion, color parsing, bounding calculations, and coroutine management.

Bertec.Utils.currentTick

Gets the current tick count in milliseconds since the application started.

void toBool(object)

Converts an object to a boolean value.

o : The object to convert.

void toBool(object , bool)

Converts an object to a boolean value, with a default fallback.

o : The object to convert.
defaultValue : The default value if conversion fails.

void toSingle(object)

Converts an object to a single-precision floating point value.

o : The object to convert.

void toSingle(object , float)

Converts an object to a single-precision floating point value, with a default fallback.

o : The object to convert.
defaultValue : The default value if conversion fails.

void toInt(object)

Converts an object to an integer value.

o : The object to convert.

void toInt(object , int)

Converts an object to an integer value, with a default fallback.

o : The object to convert.
defaultValue : The default value if conversion fails.

void toUnsigned64(object)

Converts an object to an unsigned 64-bit integer value.

o : The object to convert.

void toUnsigned64(object , ulong)

Converts an object to an unsigned 64-bit integer value, with a default fallback.

o : The object to convert.
defaultValue : The default value if conversion fails.

void toString(object)

Converts an object to a string.

o : The object to convert.

void toLowerString(object)

Converts an object to a lower-case string.

o : The object to convert.

void toCollection(object)

Converts an object to a dictionary of string/object pairs.

o : The object to convert.

void toArray(object)

Converts an object to a string array.

o : The object to convert.

void toHex(byte[])

Converts a byte array to a hexadecimal string.

bytes : The byte array.

void fromHex(string)

Converts a hexadecimal string to a byte array.

hex : The hexadecimal string.

void parseColor(string , UnityEngine.Color)

Parses a color string (hex or comma-separated) into a Unity Color.

parm : The color string.
val : The default color value.

void parseColor(string)

Parses a color string (hex or comma-separated) into a Unity Color.

parm : The color string.

void GetBoundingForGameObject(UnityEngine.GameObject , bool)

Calculates the bounds of a GameObject and its children.

g : The GameObject.
includeInactive : Whether to include inactive children.

void GetBoundingForGameObject(UnityEngine.GameObject , UnityEngine.Bounds[])

Calculates the bounding box for a GameObject given a list of child bounds.

g : The GameObject.
lst : The list of bounds.

void GetBoundsForChildren(UnityEngine.GameObject , bool)

Gets the bounding boxes for a GameObject's children (does not include itself).

g : The GameObject.
includeInactive : Whether to include inactive children.

Bertec.Utils.Coroutines

Coroutine helper methods.

void WaitForSeconds(float , bool)

Waits for the specified amount of seconds, either in scaled time (just as Unity's UnityEngine.WaitForSeconds) or in unscaled time.

seconds : Duration in seconds to wait before continuing.
unscaled : Should the wait time ignore UnityEngine.Time.timeScale?

void WaitForSecondsRealtime(float)

Waits for the specified amount of seconds in real time. Lighter replacement for UnityEngine.WaitForSecondsRealtime.

seconds : The amount of seconds to wait for.

void WaitForAsyncOperation(UnityEngine.AsyncOperation)

Waits until the specified UnityEngine.AsyncOperation is done.

operation : The async operation to wait for.

Bertec.Utils.Coroutines.WaitForEndOfFrame

Suspends a coroutine until the very end of the current frame.

void Start(Collections.IEnumerator)

Starts a new coroutine.

enumerator : The enumerator to execute.

void Start(Collections.IEnumerator , int)

Starts a new coroutine.

enumerator : The enumerator to execute.
updateLoopId : Which update loop should the coroutine be part of?

void Start(Collections.IEnumerator , Bertec.Utils.Coroutines.UpdateLoop)

Starts a new coroutine.

enumerator : The enumerator to execute.
updateLoop : Which update loop should the coroutine be part of?

void Start(Collections.IEnumerator , UnityEngine.GameObject , Bertec.Utils.Coroutines.UpdateLoop)

enumerator : The enumerator to execute.
linkedObject : Which gameobject to link the coroutine's lifetime with.
updateLoop : Which update loop should the coroutine be part of?

void Start(Collections.IEnumerator , UnityEngine.GameObject , int)

enumerator : The enumerator to execute.
linkedObject : Which gameobject to link the coroutine's lifetime with.
updateLoopId : Which update loop should the coroutine be part of?

void Stop(int)

Stops a running coroutine prematurely. This will stop any child coroutines as well.

id : The id of the coroutine to stop.

Bertec.Utils.Coroutines.StopAll()

Stops all coroutines.

void IsRunning(int)

Checks whether a coroutine with the given ID is running. A paused coroutine is still considered running.

id : The id of the coroutine to check.

void SetPaused(int , bool)

Pauses or unpauses a coroutine.

id : The id of the coroutine.
paused : True to pause, false to unpause.

void IsPaused(int)

Checks whether a coroutine is currently paused either directly or because of a paused parent.

id : Id of the coroutine.

void Pause(int)

Pauses a coroutine.

id : Id of the coroutine to pause.

void Unpause(int)

Unpauses a paused coroutine.

id : Id of the coroutine to unpause.

Bertec.VideoStreamer.Start()

Starts listening for the stream start/stop commands to be issued. Called this from VideoStreamerContainer_Impl

---

Bertec.FlowMovementSpeedData

Data representing the left and right belt speeds for visual flow movement.

Bertec.VisualFlowMovementSpeed

Handles updates to the visual flow movement speed and provides event notification.

Bertec.VisualFlowMovementSpeed.SpeedUpdated

Event triggered when the flow movement speed is updated.

data : The updated flow movement speed data.

Bertec.VisualFlowMovementSpeed.CurrentFlowSpeed

Gets the current flow movement speed data.

RuntimeConstants

Runtime constants for platform and build configuration detection.

RuntimeConstants.UrpEnabled

Indicates whether the Universal Render Pipeline (URP) is enabled.

RuntimeConstants.IL2CPP

Indicates whether the build is using IL2CPP scripting backend.

RuntimeConstants.IsWindows

Indicates whether the application is running on Windows.

RuntimeConstants.IsAndroid

Indicates whether the application is running on Android.