Entity Api

Grouping entity and component creation and handling here. More...

Classes

struct	Quantum.EntityRef Quantum entity reference. More...
class	Quantum.Core.FrameBase The Frame class is the container for all the transient and static game state data, including the API for entities, physics, assets and others. More...
struct	Quantum.Core.FrameBase.FrameBaseUnsafe Frame API to give access to C# unsafe pointers and advanced immediate operations. More...

Functions

AddResult	Quantum.Core.FrameBase.Add (EntityRef entity, int componentIndex, out void *result) Adds a component of default value to an entity and returns a pointer to the added component. More...
AddResult	Quantum.Core.FrameBase.Add (EntityRef entity, int componentIndex, void *value) Adds a component of defined value to an entity More...
AddResult	Quantum.Core.FrameBase.Add (EntityRef entity, int componentIndex, void *value, out void *result) Adds a component of defined value to an entity and returns a pointer to the added component. More...
AddResult	Quantum.Core.FrameBase.Add< T > (EntityRef entity) Adds a default component to an entity. More...
AddResult	Quantum.Core.FrameBase.Add< T > (EntityRef entity, out T *result) Adds a component of default value to an entity and returns a pointer to the added component. More...
AddResult	Quantum.Core.FrameBase.Add< T > (EntityRef entity, T value) Adds a component to an entity. More...
AddResult	Quantum.Core.FrameBase.Add< T > (EntityRef entity, T value, out T *result) Adds a component of defined value to an entity and returns a pointer to the added component. More...

bool	Quantum.Core.FrameBase.AddOrGet (EntityRef entity, int componentIndex, out void *result) Adds a component of default value to an entity (if it does not have that component yet) and gets a pointer to the component. More...
bool	Quantum.Core.FrameBase.AddOrGet< T > (EntityRef entity, out T *result) Adds a component of default value to an entity (if it does not have that component yet) and gets a pointer to the component. More...
EntityRef	Quantum.Core.FrameBase.Create () Creates an entity that is saved in the game state. More...
EntityRef	Quantum.Core.FrameBase.Create (AssetRefEntityPrototype prototype) Creates an entity from a prototype asset. This process is also reffered to as "prototype materialization". More...
EntityRef	Quantum.Core.FrameBase.Create (EntityPrototype prototype) Creates an entity from a prototype. This process is also reffered to as "prototype materialization". More...
void	Quantum.Core.FrameBase.Create (EntityPrototypeContainer[] prototypes, Map parentAsset) Creates (materializes) map prototypes. The difference between this method and calling Create(EntityPrototype) repeatedly is that EntityRef fields get resolved and that MapEntityLink components are added implicitly. More...
bool	Quantum.Core.FrameBase.Destroy (EntityRef entityRef) Destroys the entity and all components that were added to it. More...
bool	Quantum.Core.FrameBase.Exists (EntityRef entityRef) Checks if an entity is still valid. More...
T	Quantum.Core.FrameBase.Get< T > (EntityRef entityRef) Gets a component from an entity. More...
ComponentIterator< T >	Quantum.Core.FrameBase.GetComponentIterator< T > () Create a component iterator for all components of one type. More...
ComponentSet	Quantum.Core.FrameBase.GetComponentSet (EntityRef entityRef) Gets a set of all component types that were added to the entity. More...
T *	Quantum.Core.FrameBase.FrameBaseUnsafe.GetPointer< T > (EntityRef entityRef) Gets a pointer to a component that can be changed directly without writing the component back with Set<T>(EntityRef, T).

	More...
bool	Quantum.Core.FrameBase.Has (EntityRef entityRef, ComponentSet set) Checks if the entity contains a whole set of components. More...
bool	Quantum.Core.FrameBase.Remove (EntityRef entityRef, int componentIndex) Removes a component from an entity. More...
bool	Quantum.Core.FrameBase.Remove (EntityRef entityRef, Type componentType) Removes a component from an entity. More...
bool	Quantum.Core.FrameBase.Remove< T > (EntityRef entityRef) Removes a component from an entity. More...
SetResult	Quantum.Core.FrameBase.Set (EntityRef entity, EntityPrototype prototype) Adds (materializes) components to an already existing entity. If a component already exists, it will get completely overwritten, but an error message will be logged. To avoid errors in such case use Set(EntityRef, EntityPrototype, out ComponentSet) instead. More...
SetResult	Quantum.Core.FrameBase.Set (EntityRef entity, EntityPrototype prototype, out ComponentSet overwrittenComponents) Adds (materializes) components to an already existing entity. If a component already exists, it will get completely overwritten. More...
SetResult	Quantum.Core.FrameBase.Set (EntityRef entity, int componentIndex, void *value) Sets a component on an entity. More...
SetResult	Quantum.Core.FrameBase.Set< T > (EntityRef entity, T value) Sets a component on an entity. More...
bool	Quantum.Core.FrameBase.TryGet< T > (EntityRef entityRef, out T value) Gets a component from an entity. Does not throw when the component does not exist. More...
bool	Quantum.Core.FrameBase.TryGetComponentSet (EntityRef entityRef, out ComponentSet set) Gets a set of all component types that were added to the entity. More...
bool	Quantum.Core.FrameBase.FrameBaseUnsafe.TryGetPointer< T > (EntityRef entityRef, out T *value) Similar to GetPointer<T>(EntityRef) but does not throw an exception if the component is not present. More...

Detailed Description

Grouping entity and component creation and handling here.

Function Documentation

◆ Add< T >() [1/4]

AddResult Quantum.Core.FrameBase.Add< T > (EntityRef entity) inline

Adds a default component to an entity.

Template Parameters

T Component type

Parameters

entity Entity reference

Internally calls Has<T>(EntityRef) and Set<T>(EntityRef, T)

Exceptions

InvalidOperationException Thrown when the entity does not exist

InvalidOperationException Thrown when entity already has a component of the given type

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ Has()

bool Quantum.Core.FrameBase.Has (EntityRef entityRef,
ComponentSet set
) inline

Checks if the entity contains a whole set of components.

Parameters

entityRef Entity reference

set Set of component types

Returns

True if the entity has all component types

◆ Exists()

bool Quantum.Core.FrameBase.Exists (EntityRef entityRef) inline

Checks if an entity is still valid.

For example handy for when you store the entity as a reference inside other components.

Parameters

entityRef Entity reference

Returns

True if the entity exists

◆ GetComponentIterator< T >()

ComponentIterator<T> Quantum.Core.FrameBase.GetComponentIterator< T >( ) inline

Create a component iterator for all components of one type.

Template Parameters

T Component type

Returns

Component iterator

Changed components have to be Set<T>(EntityRef, T) back.

Accessing the component array creates a copy of the component.

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ Add< T >() [2/4]

AddResult Quantum.Core.FrameBase.Add< T > (EntityRef entity, T value) inline

Adds a component to an entity.

Template Parameters

T Component type

Parameters

entity Entity reference

value Value of the component to be added

Internally calls Has<T>(EntityRef) and Set<T>(EntityRef, T)

Returns

AddResult.EntityDoesNotExist if *entity* does not exist. AddResult.ComponentAlreadyExists if the *entity* already has a component of type *T*. The value is not set in this case. AddResult.ComponentAdded Otherwise, indicating that the component was successfully added to the entity.

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ Add< T >() [3/4]

AddResult Quantum.Core.FrameBase.Add< T > (EntityRef entity,
out T * result
)

inline

Adds a component of default value to an entity and returns a pointer to the added component.

Template Parameters

T Component type

Parameters

entity Entity reference

result A pointer to the component added. Null if the result is not AddResult.ComponentAdded.

Internally calls Has<T>(EntityRef) and Set<T>(EntityRef, T)

Returns

AddResult.EntityDoesNotExist if *entity* does not exist. AddResult.ComponentAlreadyExists if the *entity* already has a component of type *T* . The value is not set in this case. AddResult.ComponentAdded Otherwise, indicating that the component was successfully added to the entity.

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ Add< T >() [4/4]

```
AddResult Quantum.Core.FrameBase.Add< T > ( EntityRef  entity,
                                            T          value,
                                            out T *    result
                                          )                          inline
```

Adds a component of defined value to an entity and returns a pointer to the added component.

Template Parameters

T Component type

Parameters

entity Entity reference

value Value of the component to be added

result A pointer to the component added. Null if the result is not AddResult.ComponentAdded.

Internally calls Has<T>(EntityRef) and Set<T>(EntityRef, T)

Returns

AddResult.EntityDoesNotExist if *entity* does not exist. AddResult.ComponentAlreadyExists if the *entity* already has a component of type *T*. The value is not set in this case. AddResult.ComponentAdded Otherwise, indicating that the component was successfully added to the entity.

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ Add() [1/3]

```
AddResult Quantum.Core.FrameBase.Add ( EntityRef entity,
                                       int       componentIndex,
                                       void *    value
                                     )                                  inline
```

Adds a component of defined value to an entity

Parameters

entity	Entity reference
componentIndex	The index of the component to be added. See also ComponentTypeId.GetComponentIndex(Type) or ComponentTypeId<T>.Id to retrieve the index of a component type.
value	A pointer to data that should be copied as value to the component added. If null, the component is added with default value.

Internally calls Has<T>(EntityRef) and Set<T>(EntityRef, T)

Returns

AddResult.EntityDoesNotExist if *entity* does not exist.
AddResult.ComponentAlreadyExists if the *entity* already has a component of index *componentIndex* . The value is not set in this case.
AddResult.ComponentAdded Otherwise, indicating that the component was successfully added to the entity.

◆ Add() [2/3]

```
AddResult Quantum.Core.FrameBase.Add ( EntityRef  entity,
                                       int        componentIndex,
                                       out void * result
                                     )                                  inline
```

Adds a component of default value to an entity and returns a pointer to the added component.

Parameters

entity	Entity reference
componentIndex	The index of the component to be added. See also ComponentTypeId.GetComponentIndex(Type) or ComponentTypeId<T>.Id to retrieve the index of a component type.
result	A pointer to the component added. Null if the result is not AddResult.ComponentAdded.

Internally calls Has<T>(EntityRef) and Set<T>(EntityRef, T)

Returns

AddResult.EntityDoesNotExist if *entity* does not exist.
AddResult.ComponentAlreadyExists if the *entity* already has a component of index *componentIndex* . The value is not set in this case.
AddResult.ComponentAdded Otherwise, indicating that the component was successfully added to the entity.

◆ Add() [3/3]

```
AddResult Quantum.Core.FrameBase.Add ( EntityRef  entity,
                                       int        componentIndex,
                                       void *     value,
                                       out void * result
                                     )                                  inline
```

Adds a component of defined value to an entity and returns a pointer to the added component.

Parameters

entity	Entity reference
componentIndex	The index of the component to be added. See also ComponentTypeId.GetComponentIndex(Type) or ComponentTypeId<T>.Id to retrieve the index of a component type.
value	A pointer to data that should be copied as value to the component added. If null, the component is added with default value.
result	A pointer to the component added. Null if the result is not AddResult.ComponentAdded.

Internally calls Has<T>(EntityRef) and Set<T>(EntityRef, T)

Exceptions

IndexOutOfRangeException	If the *componentIndex* does not identify a valid component.

Returns

AddResult.EntityDoesNotExist if *entity* does not exist.
AddResult.ComponentAlreadyExists if the *entity* already has a component of index *componentIndex* . The value is not set in this case.
AddResult.ComponentAdded Otherwise, indicating that the component was successfully added to the entity.

◆ AddOrGet< T >()

bool Quantum.Core.FrameBase.AddOrGet< T > (EntityRef entity,
out T * result
)
inline

Adds a component of default value to an entity (if it does not have that component yet) and gets a pointer to the component.

Parameters

entity Entity reference

result A pointer to the component added or already existing. Null if the *entity* does not exist.

Returns

False if the entity does not exist and True otherwise.

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ AddOrGet()

```
bool Quantum.Core.FrameBase.AddOrGet ( EntityRef  entity,
                                       int        componentIndex,
                                       out void * result
                                     )                                  inline
```

Adds a component of default value to an entity (if it does not have that component yet) and gets a pointer to the component.

Parameters

entity	Entity reference
componentIndex	The index of the component to be added. See also ComponentTypeId.GetComponentIndex(Type) or ComponentTypeId<T>.Id to retrieve the index of a component type.
result	A pointer to the component added or already existing. Null if the *entity* does not exist.

Exceptions

IndexOutOfRangeException	If the *componentIndex* does not identify a valid component.

Returns

False if the entity does not exist and True otherwise.

◆ Set< T >()

SetResult Quantum.Core.FrameBase.Set< T > (EntityRef entity, T value) inline

Sets a component on an entity.

Template Parameters

T Component type

Parameters

entity Entity ref

value Value of the component to be added

Returns

SetResult.EntityDoesNotExist if *entity* does not exist. SetResult.ComponentUpdated if the *entity* already had a component of type *T* that had is value updated to *value* . SetResult.ComponentAdded Otherwise, indicating that a component with the defined value was added to the entity.

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ Set() [1/3]

SetResult Quantum.Core.FrameBase.Set (EntityRef entity,
int componentIndex,
void * value
) inline

Sets a component on an entity.

Parameters

entity	Entity ref
componentIndex	The index of the component being set. See also ComponentTypeId.GetComponentIndex(Type) or ComponentTypeId<T>.Id to get the index of a component.
value	A pointer to data that should be copied as value to the component. If null, the component is set with default value.

Exceptions

IndexOutOfRangeException	If the *componentIndex* does not identify a valid component.

Returns

SetResult.EntityDoesNotExist if *entity* does not exist. SetResult.ComponentUpdated if the *entity* already had a component of index *componentIndex* that had is value updated to *value* . SetResult.ComponentAdded Otherwise, indicating that a component with the defined value was added to the entity.

◆ Get< T >()

T Quantum.Core.FrameBase.Get< T > (EntityRef entityRef) inline

Gets a component from an entity.

Template Parameters

T Component type

Parameters

entityRef Entity reference

Returns

Requested component

Modified components need to be explicitly written back by using Set<T>(EntityRef, T).

Use Has<T>(EntityRef) to quickly check the entity for component availability.

```
var t = f.Get<Transform2D>(entity);
t.Position = FPVector2.Zero;
f.Set(entity, t);
```

Exceptions

InvalidOperationException Thrown when the entity does not exist

InvalidOperationException Thrown when the component does not exists on entity

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ TryGet< T >()

bool Quantum.Core.FrameBase.TryGet< T > (EntityRef entityRef,
out T value
)

inline

Gets a component from an entity. Does not throw when the component does not exist.

Template Parameters

T Component type

Parameters

entityRef Entity reference

value Requested component

Returns

True if the component exists

```
if (f.TryGet<Transform2D>(entity, out Transform2D t)) {
  t.Position = FPVector2.Zero;
  f.Set(entity, t);
}
```

Exceptions

InvalidOperationException Thrown when the entity does not exist

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ Remove< T >()

bool Quantum.Core.FrameBase.Remove< T > (EntityRef entityRef) inline

Removes a component from an entity.

Template Parameters

T Component type

Parameters

entityRef Entity reference

Returns

False if the *entityRef* is not valid or the entity does not have a component of type *T* . True otherwise.

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ Remove() [1/2]

bool Quantum.Core.FrameBase.Remove (EntityRef entityRef,
Type componentType
) inline

Removes a component from an entity.

Parameters

entityRef Entity reference

componentType Component type

Returns

False if the *entityRef* is not valid or the entity does not have a component of type *componentType* . True otherwise.

◆ Remove() [2/2]

bool Quantum.Core.FrameBase.Remove (EntityRef entityRef,
int componentIndex
) inline

Removes a component from an entity.

Parameters

entityRef Entity reference

componentIndex The index of the component to be removed.

Exceptions

IndexOutOfRangeException If the *componentIndex* does not identify a valid component.

Returns

False if the *entityRef* is not valid or the entity does not have a component of index *componentIndex* . True otherwise.

◆ GetComponentSet()

ComponentSet
Quantum.Core.FrameBase.GetComponentSet (EntityRef entityRef) inline

Gets a set of all component types that were added to the entity.

Parameters

entityRef Entity reference

Returns

A set of all component types that are available on the entity

◆ TryGetComponentSet()

bool Quantum.Core.FrameBase.TryGetComponentSet (EntityRef entityRef, out ComponentSet set) inline

Gets a set of all component types that were added to the entity.

Parameters

entityRef Entity reference

set

◆ Destroy()

bool Quantum.Core.FrameBase.Destroy (EntityRef entityRef) inline

Destroys the entity and all components that were added to it.

The destruction takes immediate effect, i.e., Exists will start to return false. The old data, however, will remain in memory until destroy/remove commands are committed, according to the CommitCommandsMode or manually through FrameBaseUnsafe.CommitAllCommands().

```
f.Destroy(entity);
```

Parameters

entityRef

◆ Create() [1/4]

EntityRef Quantum.Core.FrameBase.Create (AssetRefEntityPrototype prototype) inline

Creates an entity from a prototype asset. This process is also reffered to as "prototype materialization".

Parameters

prototype

See also

Create(EntityPrototype)

Returns

◆ Create() [2/4]

EntityRef Quantum.Core.FrameBase.Create (EntityPrototype prototype) inline

Creates an entity from a prototype. This process is also reffered to as "prototype materialization".

The steps are following:

1. An empty entity is created.
2. For each component prototype:
   a. Component prototype is materialized.
   b. Component prototype has MaterializeUser invoked.
   c. Component prototype is added to the entity.
3. Quantum.ISignalOnEntityPrototypeMaterialized is invoked.

Parameters

prototype

Returns

◆ Create() [3/4]

void Quantum.Core.FrameBase.Create (EntityPrototypeContainer[] prototypes, Map parentAsset) inline

Creates (materializes) map prototypes. The difference between this method and calling Create(EntityPrototype) repeatedly is that EntityRef fields get resolved and that MapEntityLink components are added implicitly.

This is a multi-step process:

1. An empty entity is created for each prototype
2. For each entity-prototype pair:
    a. For each component prototype:
        i. Component prototype is materialized.
        ii. Component prototype has MaterializeUser invoked.
        iii. Component prototype is added to the entity.
    b. MapEntityLink component is added to the entity.
3. ISignalOnEntityPrototypeMaterialized is invoked for each entity-prototype pair.

Parameters

prototypes

parentAsset

◆ Set() [2/3]

SetResult Quantum.Core.FrameBase.Set (EntityRef entity, EntityPrototype prototype) inline

Adds (materializes) components to an already existing entity. If a component already exists, it will get completely overwritten, but an error message will be logged. To avoid errors in such case use Set(EntityRef, EntityPrototype, out ComponentSet) instead.

The steps are following:

1. For each component prototype:
    a. Component prototype is materialized.
    b. Component prototype has MaterializeUser invoked.
    c. Component prototype is added to the entity.
2. Quantum.ISignalOnEntityPrototypeMaterialized is invoked.

Parameters

entity

prototype

Returns

SetResult.EntityDoesNotExist if entity does not exist.
SetResult.ComponentUpdated if any component has been overwritten,
SetResult.ComponentAdded otherwise.

See also

Create(EntityPrototype), Set(EntityRef, EntityPrototype, out ComponentSet, out ComponentSet)

◆ Set() [3/3]

```
SetResult
Quantum.Core.FrameBase.Set    ( EntityRef       entity,
                                EntityPrototype prototype,
                                out ComponentSet overwrittenComponents
                              )
```
inline

Adds (materializes) components to an already existing entity. If a component already exists, it will get completely overwritten.

The steps are following:

1. For each component prototype:
    a. Component prototype is materialized.
    b. Component prototype has MaterializeUser invoked.
    c. Component prototype is added to the entity.
2. Quantum.ISignalOnEntityPrototypeMaterialized is invoked.

Parameters

entity

prototype

overwrittenComponents Components that have been overwritten.

Returns

SetResult.EntityDoesNotExist if entity does not exist.
SetResult.ComponentUpdated if any component has been overwritten,
SetResult.ComponentAdded otherwise.

See also

Create(EntityPrototype)

◆ Create() [4/4]

```
EntityRef Quantum.Core.FrameBase.Create ( )
```
inline

Creates an entity that is saved in the game state.

Returns

Entity reference

```
var entity = f.Create();
```

◆ GetPointer< T >()

T*
Quantum.Core.FrameBase.FrameBaseUnsafe.GetPointer<
T > **(EntityRef entityRef)** inline

Gets a pointer to a component that can be changed directly without writing the component back with Set<T>(EntityRef, T).

Template Parameters

T Component type

Parameters

entityRef Entity reference

Returns

Pointer to the requested component

Exceptions

InvalidOperationException Thrown when the entity does not exist

InvalidOperationException Thrown when the entity does have the requested component type.

Always check the availability of the component with Has(EntityRef, ComponentSet) before using this method.

Type Constraints

T : *unmanaged*

T : *IComponent*

◆ TryGetPointer< T >()

bool Quantum.Core.FrameBase.FrameBaseUnsafe.TryGetPointer<T>(EntityRef entityRef, out T * value) inline

Similar to GetPointer<T>(EntityRef) but does not throw an exception if the component is not present.

Template Parameters

T Component type

Parameters

entityRef Entity reference

value Resulting component pointer of null if the component was not found

Returns

True if component was found

Exceptions

InvalidOperationException Thrown when the entity does not exist

Type Constraints

T : *unmanaged*

T : *IComponent*