CORBA Application Programming Interface (API)

25 November 1997

Bionic Buffalo Corporation

2533 North Carson Street, Suite 1884
Carson City, Nevada 89706-0147 USA

http://www.tatanka.com

©1997 Bionic Buffalo Corporation; All Rights Reserved. This document contains licensed information which is the property of Bionic Buffalo Corporation, and is not to be duplicated without written permission.

Document Revision History

Original release, 5 July 1997.

Update, 2 September 1997.

Update, 25 November 1997: preliminary draft for portable object adaptor.

Printed 10:02:44, Monday, 18 May, 1998.

Additional Document Information

Project name: *egypt*.

Document file name: *egapi02.doc*.

Written with Microsoft Word 6. Illustrations created with Visio Technical 4.5. Set in Monotype Bulmer, Adobe Ocean Sans, and Adobe Tekton.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Contents

©1997 Bionic Buffalo Corporation; All Rights Reserved.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Application Programming Interface (API)

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AliasfDef::absolute_name - get scoped name of alias definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::absolute_name

CORBA::AliasDef::containing_repository - get repository of alias definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::containing_repository

CORBA::AliasDef::def_kind - get definition kind of a alias definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::AliasDef::defined_in - get container of alias definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AliasDef::describe - describe alias definition

Inherited from

CORBA::TypedefDef

Refer to

CORBA::TypedefDef::describe

CORBA::AliasDef::destroy - destroy an alias definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::AliasDef::id - global identifier of alias definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

CORBA::AliasDef::move - move alias definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AliasDef::name - local identifier of alias definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AliasDef::original_type_def - basis type of alias definition

Synopsis - C

```
CORBA_IDLType     CORBA_AliasDef__get_original_type_def
                                ( CORBA_AliasDef        aliasdef1,
                                  CORBA_Environment     * ev1 ) ;

void              CORBA_AliasDef__set_original_type_def
                                ( CORBA_AliasDef        aliasdef1,
                                  CORBA_IDLType         typedef1,
                                  CORBA_Environment     * ev1 ) ;
```

Arguments

constantdef1	*(in)* the constant definition object
typedef1	*(in)* the new IDL type being aliased
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_AliasDef__get_original_type_def(): the IDL type which is aliased
CORBA_AliasDef__set_orignal_type_def(): *(void)*

Exceptions

(standard)

Description

Return or set the IDL type object underlying an alias definition object.

Notes

(none)

Reference

CORBA, 6.5.13.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AliasDef::type - type of an alias definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

CORBA::AliasDef::version - get version of an alias definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

CORBA::ArrayDef::def_kind - get definition kind of an array definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::ArrayDef::destroy - destroy an array definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ArrayDef::element_type - type code of sequence element

Synopsis - C

```
CORBA_TypeCode          CORBA_ArrayDef__get_element_type
                            ( CORBA_ArrayDef        arraydef1,
                            CORBA_Environment       * ev1 ) ;
```

Arguments

arraydef1	*(in)* the array definition object
ev1	*(in/out)* the CORBA Environment

Returns

the typecode of the array element

Exceptions

(standard)

Description

Return the type code of the array element.

Notes

1. This attribute is ***readonly*** and cannot be set. However, it can be changed indirectly by setting the ***element_type_def*** attribute.

Reference

CORBA, 6.5.18.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ArrayDef::element_type_def - IDL type of sequence element

Synopsis - C

```
CORBA_IDLType       CORBA_ArrayDef__get_element_type_def
                        ( CORBA_ArrayDef        arraydef1,
                        CORBA_Environment       * ev1 ) ;

void                CORBA_ArrayDef__set_element_type_def
                        ( CORBA_ArrayDef        arraydef1,
                        CORBA_TypeDef           element1
                        CORBA_Environment       * ev1 ) ;
```

Arguments

arraydef1	*(in)* the array definition object
element1	*(in)* the array element IDL type object
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_ArrayDef__get_element_type_def(): the array element IDL type
CORBA_ArrayDef__set_element_type_def(): *(void)*

Exceptions

(standard)

Description

Returns or sets the IDL type of an array element.

Notes

1. Setting this attribute will automatically set the ***element_type*** and ***type*** attributes.

Reference

CORBA, 6.5.18.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ArrayDef::length - number of elements in an array

Synopsis - C

```
CORBA_unsigned_long     CORBA_ArrayDef__get_length
                            ( CORBA_ArrayDef        arraydef1,
                            CORBA_Environment       * ev1 ) ;

void                    CORBA_ArrayDef__set_length
                            ( CORBA_ArrayDef        arraydef1,
                            CORBA_unsigned_long     length1
                            CORBA_Environment       * ev1 ) ;
```

Arguments

arraydef1	*(in)* the array definition object
length1	*(in)* the number of elements in the array
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_ArrayDef__get_length(): the number of elements in the array
CORBA_ArrayDef__set_length(): *(void)*

Exceptions

(standard)

Description

Returns or sets the number of elements in an array.

Notes

1. Setting this attribute will automatically set the ***type*** attribute.

Reference

CORBA, 6.5.18.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ArrayDef::type - type of an array definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

CORBA::AttributeDef::absolute_name - get scoped name of an attribute definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::absolute_name

CORBA::AttributeDef::containing_repository - get repository of an attribute definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::containing_repository

CORBA::AttributeDef::def_kind - get definition kind of an attribute definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AttributeDef::defined_in - get container of attribute definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AttributeDef::describe - describe attribute definition

Synopsis - C

```
typedef struct
        {
        CORBA_DefinitionKind        kind ;
        CORBA_any                   value ;
        }
        CORBA_Contained_Description ;

typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_RepositoryId          id ;
        CORBA_RepositoryId          defined_in ;
        CORBA_VersionSpec           version ;
        CORBA_TypeCode              type ;
        CORBA_AttributeMode         mode ;
        }
        CORBA_AttributeDescription ;

CORBA_Contained_Description     CORBA_AttributeDef_describe
                                        ( CORBA_AttributeDef    attributedef1,
                                        CORBA_Environment       * ev1 ) ;
```

Arguments

attributedef1	*(in)* the attribute definition object
ev1	*(in/out)* the CORBA Environment

Returns

a description of the attribute definition object

Exceptions

(standard)

Description

Returns a description of an attribute definition object. The ***value*** within the description is the ***CORBA::AttributeDescription*** data structure.

Notes

1. Inherited from ***CORBA::Contained***. Other interfaces which inherit from ***CORBA::Contained*** may return different structures for the ***value*** element of ***CORBA::Contained::Description***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

CORBA, 6.5.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AttributeDef::destroy - destroy an attribute definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::AttributeDef::id - global identifier of an attribute definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AttributeDef::mode - mode of attribute definition

Synopsis - C

```
CORBA_AttributeMode  CORBA_AttributeDef__get_mode
                                ( CORBA_AttributeDef      attributedef1,
                                  CORBA_Environment       * ev1 ) ;
void                 CORBA_AttributeDef__set_mode
                                ( CORBA_AttributeDef      attributedef1,
                                  CORBA_AttributeMode     mode1,
                                  CORBA_Environment       * ev1 ) ;
```

Arguments

attributedef1	*(in)* the attribute definition object
mode1	*(in)* the new mode (either ***CORBA_ATTR_NORMAL*** or ***CORBA_ATTR_READONLY***)
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_AttributeDef__get_mode(): the mode of the attribute definition object (either ***CORBA_ATTR_NORMAL*** or ***CORBA_ATTR_READONLY***)
CORBA_AttributeDef__set_mode(): *(void)*

Exceptions

(standard)

Description

Return or set the mode of an attribute definition object.

Notes

(none)

Reference

CORBA, 6.5.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AttributeDef::move - move attribute definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::AttributeDef::name - local identifier of an attribute definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AttributeDef::type - type code of attribute definition

Synopsis - C

```
CORBA_TypeCode      CORBA_AttributeDef__get_type
                        ( CORBA_AttributeDef      attributedef1,
                        CORBA_Environment         * ev1 ) ;
```

Arguments

attributedef1	*(in)* the attribute definition object
ev1	*(in/out)* the CORBA Environment

Returns

the type code of the attribute definition object

Exceptions

(standard)

Description

Returns the type code of an attribute definition object.

Notes

1. Although the ***CORBA::AttributeDef::type*** attribute is ***readonly***, it can be modified indirectly by setting the ***CORBA::AttributeDef::type_def*** attribute (*q.v.*).

Reference

CORBA, 6.5.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AttributeDef::type_def - IDL type of attribute definition

Synopsis - C

```
CORBA_IDLType   CORBA_AttributeDef__get_type_def
                        ( CORBA_AttributeDef    attributedef1,
                        CORBA_Environment       * ev1 ) ;
void            CORBA_AttributeDef__set_type_def
                        ( CORBA_AttributeDef    attributedef1,
                        CORBA_IDLType           typedef1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

attributedef1	*(in)* the attribute definition object
typedef1	*(in)* the new IDL type object
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_AttributeDef__get_type_def(): the IDL type object of the attribute definition object

CORBA_AttributeDef__set_type_def(): *(void)*

Exceptions

(standard)

Description

Return or set the IDL type object of an attribute definition object.

Notes

1. Setting the ***type_def*** attribute causes the ***type*** attribute to be modified.

Reference

CORBA, 6.5.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::AttributeDef::version - get version of an attribute definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::change_implementation - change the implementation of an object

Synopsis - C

```
typedef CORBA_Object            ImplementationDef ;

void    CORBA_BOA_change_implementation ( CORBA_BOA boa1,
                                CORBA_Object obj1, CORBA_ImplementationDef impl1,
                                CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor to create the object
obj1	(in) the object whose implementation is to be changed
impl1	(in) the new ImplementationDef of the object
ev1	(in/out) the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Changes the ***ImplementationDef*** of an existing object.

Notes

1. The old ***ImplementationDef*** is not destroyed or changed.

Reference

CORBA, 8.2.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::create - create an object

Synopsis - C

```
typedef struct
        {
        CORBA_unsigned_long  _maximum,
        CORBA_unsigned_long  _length,
        CORBA_octet          *_buffer
        }
        CORBA_ReferenceData ;
typedef CORBA_Object         InterfaceDef ;
typedef CORBA_Object         ImplementationDef ;

CORBA_Object  CORBA_BOA_create ( CORBA_BOA boa1, CORBA_ReferenceData * id1,
                        CORBA_InterfaceDef intf1, CORBA_ImplementationDef impl1,
                        CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor to create the object
ref1	(in) private data to belong to the new object
intf1	(in) the InterfaceDef of the new object
impl1	(in) the ImplementationDef of the new object
ev1	(in/out) the CORBA Environment

Returns

an object reference for the new object

Exceptions

(standard)

Description

Creates an object from basic components.

Notes

1. The ***InterfaceDef*** is produced by the IDL compiler.

2. The ***ImplementationDef*** is returned by eg_ImplementationDef_create().

3. Through ***CORBA::BOA::change_implementation***, the ***ImplementationDef*** can be changed after the object is created. However, there is no mechanism for modification of the ***InterfaceDef*** or the ***ReferenceData*** once the object is created.

4. Although some CORBA implementations use a string containing the IDL for an object as the object's InterfaceDef, this is not a portable usage. The form of ***InterfaceDef*** is implementation-dependent.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

CORBA, 8.2.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::deactivate_impl - inform that an implementation is no longer active

Synopsis - C

```
typedef CORBA_Object   CORBA_ImplementationDef ;

void     CORBA_BOA_deactivate_impl ( CORBA_BOA boa1,
                                     CORBA_ImplementationDef impl1,
                                     CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor which owns the implementation
impl1	(in) the implementation to be deactivated
ev1	(in/out) the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Used by an implementation under a shared server or persistent server activation policy to notify the object adaptor that an implementation is no longer active.

Notes

(none)

Reference

CORBA, 8.2.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::deactivate_obj - notify object adaptor an object is no longer active

Synopsis - C

```
void    CORBA_BOA_deactivate_obj ( CORBA_BOA boa1,
                                   CORBA_Object obj1,
                                   CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor which activated the object
obj1	(in) the object which is no longer active
ev1	(in/out) the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Used by an object or implementation to notify the ***BOA*** it is no longer active.

Notes

(none)

Reference

CORBA, 8.2.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::dispose - destroy an object

Synopsis - C

```
void    CORBA_BOA_dispose ( CORBA_BOA boa1, CORBA_Object obj1,
                            CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor which created the object
obj1	(in) the object to be destroyed
ev1	(in/out) the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Destroys an object.

Notes

1. After this call, the ORB core and BOA will act as if the object has never been created, and attempts to issue requests on any existing object references for that object will fail.

2. The implementation is responsible for deallocating all resources for the object.

Reference

CORBA, 8.2.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::get_id - return the ReferenceData for an object

Synopsis - C

```
typedef struct
        {
        CORBA_unsigned_long   _maximum,
        CORBA_unsigned_long   _length,
        CORBA_octet           *_buffer
        }
        CORBA_ReferenceData ;

CORBA_ReferenceData   * CORBA_BOA_get_id ( CORBA_BOA boa1,
                                CORBA_Object obj1, CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor to create the object
obj1	(in) the object whose id is sought
ev1	(in/out) the CORBA Environment

Returns

a copy of the reference data for an bject

Exceptions

(standard)

Description

Creates an object from basic components.

Notes

1. The ***InterfaceDef*** is produced by the IDL compiler.
2. The ***ImplementationDef*** is returned by eg_ImplementationDef_create().

Reference

CORBA, 8.2.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::get_principal - determine the Principal of a request

Synopsis - C

```
typedef CORBA_Object   CORBA_Principal ;

CORBA_Principal        CORBA_BOA_get_principal ( CORBA_BOA boa1,
                              CORBA_Object obj1,
                              CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor of the implementation
obj1	(in) the object whose requesting principal is to be discovered
ev1	(in/out) the CORBA Environment

Returns

the principal initiating the current request to ***obj1***

Exceptions

(standard)

Description

Determines the principal making a request.

Notes

1. The meaning and interpretation of the principal depend on the security environment.

2. The implementation can use this call to enforce access rights to objects, perform accounting or logging operations, or for any other purpose.

Reference

CORBA, 8.2.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::impl_is_ready - notify the BOA that an implementation is ready

Synopsis - C

```
void    CORBA_BOA_impl_is_ready ( CORBA_BOA boa1,
                                  CORBA_ImplementationDef impl1,
                                  CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor which owns the implementation
impl1	(in) the implementation which is ready
ev1	(in/out) the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Notify the ***BOA*** that an implementation is ready to accept requests.

Notes

1. Used by an implementation in shared server or persistent server activation policies to notify the ***BOA*** that the implementation is active and ready to accept requests.

Reference

CORBA, 8.2.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::obj_is_ready - tell object adaptor that an object is activated

Synopsis - C

```
typedef CORBA_Object   CORBA_ImplementationDef ;

void     CORBA_BOA_obj_is_ready ( CORBA_BOA boa1,
                                  CORBA_Object obj1,
                                  CORBA_ImplementationDef impl1,
                                  CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor owning the implementation
obj1	(in) the object which is ready to receive requests
impl1	(in) the implementation owning ***obj1***
ev1	(in/out) the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Used by an implementation in a shared or unshared activation policy to notify the object adaptor that a particular object has been activated.

Notes

(none)

Reference

CORBA, 8.2.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::BOA::set_exception - raise an exception condition

Synopsis - C

```
void    CORBA_BOA_set_exception ( CORBA_BOA boa1, CORBA_exception_type major1,
                                  CORBA_string exception_name1, void * param1,
                                  CORBA_Environment * ev1 ) ;
```

Arguments

boa1	(in) the basic object adaptor belonging to the implementation
major1	(in) NO, USER, or SYSTEM exception type
exception_name1	(in) string name of the exception
param1	(in) the exception data structure
ev1	(in/out) the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Used by an implementation to set an exception condition.

Notes

1. The exception string is produced by the IDL compiler.
2. Exceptions are kept in the environment in a system-dependent fashion.

Reference

CORBA, 8.2 and 14.25.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ConstantDef::absolute_name - get scoped name of constant definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::absolute_name

CORBA::ConstantDef::containing_repository - get repository of constant definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::containing_repository

CORBA::ConstantDef::def_kind - get definition kind of a constant definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::ConstantDef::defined_in - get container of constant definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ConstantDef::describe - describe constant definition

Synopsis - C

```
typedef struct
        {
        CORBA_DefinitionKind            kind ;
        CORBA_any                       value ;
        }
        CORBA_Contained_Description ;

typedef struct
        {
        CORBA_Identifier                name ;
        CORBA_RepositoryId              id ;
        CORBA_RepositoryId              defined_in ;
        CORBA_VersionSpec               version ;
        CORBA_TypeCode                  type ;
        CORBA_any                       value ;
        }
        CORBA_ConstantDescription ;

CORBA_Contained_Description     CORBA_ConstantDef_describe
                                        ( CORBA_ConstantDef     constantdef1,
                                        CORBA_Environment       * ev1 ) ;
```

Arguments

constantdef1	*(in)* the constant definition object
ev1	*(in/out)* the CORBA Environment

Returns

a description of the constant definition object

Exceptions

(standard)

Description

Returns a description of a constant definition object. The ***value*** within the description is the ***CORBA::ConstantDescription*** data structure.

Notes

1. Inherited from ***CORBA::Contained***. Other interfaces which inherit from ***CORBA::Contained*** may return different structures for the ***value*** element of ***CORBA::Contained::Description***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

CORBA, 6.5.8.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ConstantDef::destroy - destroy a constant definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::ConstantDef::id - global identifier of a constant definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

CORBA::ConstantDef::move - move constand defintion to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::ConstantDef::name - local identifier of a constant definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ConstantDef::type - type code of a constant definition

Synopsis - C

```
CORBA_TypeCode      CORBA_ConstantDef__get_type
                        ( CORBA_ConstantDef      constantdef1,
                        CORBA_Environment        * ev1 ) ;
```

Arguments

constantdef1	*(in)* the constant definition object
ev1	*(in/out)* the CORBA Environment

Returns

the type code of the constant definition object

Exceptions

(standard)

Description

Returns the type code of a constant definition object.

Notes

1. The type of a constant must be a simple (scalar) type.

2. Although the ***CORBA::ConstantDef::type*** attribute is ***readonly***, it can be modified indirectly by setting the ***CORBA::ConstantDef::type_def*** attribute (*q.v.*).

Reference

CORBA, 6.5.8.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ConstantDef::type_def - IDL type of a constant definition

Synopsis - C

```
CORBA_IDLType     CORBA_ConstantDef__get_type_def
                                ( CORBA_ConstantDef     constantdef1,
                                  CORBA_Environment     * ev1 ) ;
void              CORBA_ConstantDef__set_type_def
                                ( CORBA_ConstantDef     constantdef1,
                                  CORBA_IDLType         typedef1,
                                  CORBA_Environment     * ev1 ) ;
```

Arguments

constantdef1	*(in)* the constant definition object
typedef1	*(in)* the new IDL type object
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_ConstantDef__get_type_def(): the IDL type object of the constant definition object

CORBA_ConstantDef__set_type_def(): *(void)*

Exceptions

(standard)

Description

Return or set the IDL type object of a constant definition object.

Notes

1. The type of a constant must be a simple (scalar) type.
2. Setting the ***type_def*** attribute causes the ***type*** attribute to be modified.

Reference

CORBA, 6.5.8.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ConstantDef::value - value of a constant definition

Synopsis - C

```
CORBA_any       CORBA_ConstantDef__get_value
                        ( CORBA_ConstantDef     constantdef1,
                        CORBA_Environment       * ev1 ) ;
void            CORBA_ConstantDef__set_value
                        ( CORBA_ConstantDef     constantdef1,
                        CORBA_any               value1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

constantdef1	*(in)* the constant definition object
value1	*(in)* the new constant value
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_ConstantDef__get_value(): the value fo the constant
CORBA_ConstantDef__set_value(): *(void)*

Exceptions

(standard)

Description

Return or set the value of a constant definition object.

Notes

1. The type of a constant must be a simple (scalar) type.
2. Setting the ***type_def*** attribute causes the ***type*** attribute to be modified.
3. The type of the ***CORBA_any*** value must match the ***type*** attribute of the ***ConstantDef***.

Reference

CORBA, 6.5.8.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ConstantDef::version - get version of a constant definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

CORBA::ConstructionPolicy::copy - copy a construction policy

Inherited from

CORBA::Policy

Refer to

CORBA::Policy::copy

CORBA::ConstructionPolicy::destroy - destroy a construction policy

Inherited from

CORBA::Policy

Refer to

CORBA::Policy::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ConstructionPolicy::make_domain_manager - set construction policy

Synopsis - C

```
void    CORBA_ConstructionPolicy_make_domain_manager
            ( CORBA_ConstructionPolicy    construction_policy1,
            CORBA_InterfaceDef            object_type1,
            CORBA_boolean                 create_new_manager1,
            CORBA_Environment             * env1 ) ;
```

Arguments

construction_policy1	*(in)* the construction policy object for a domain
object_type1	*(in)* types of object to which the new value applies
create_new_manager1	*(in)* if ***TRUE***, then a new domain manager will be created for each new object of ***object_type1***; if ***FALSE***, then new objects of ***object_type1*** will be in the current domain
env1	*(in/out)* the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Specifies whether new domain managers are to be created for new objects.

Notes

1. If new domain managers are not created, then the default domain is the domain of the creator.

2. If a new domain manager is created for a new object, then it can be retrieved using ***CORBA::Object::get_domain_managers***.

Reference

CORBA 2.1, 5.9.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ConstructionPolicy::policy_type - type of a construction policy

Inherited from

CORBA::Policy

Refer to

CORBA::Policy::policy_type

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::absolute_name - get scoped name of contained object

Synopsis - C

```
typedef CORBA_string    CORBA_ScopedName ;

CORBA_ScopedName        CORBA_Contained__get_absolute_name
                            ( CORBA_Contained           contained1,
                            CORBA_Environment           * ev1 ) ;
```

Arguments

contained1	*(in)* the contained object
ev1	*(in/out)* the CORBA Environment

Returns

the absolute name of ***contained1***

Exceptions

(standard)

Description

Returns the absolute or scoped name of the contained object.

Notes

1. The absolute or scoped name of ***contained1*** is of the form

   :: repository1 [:: container1 [:: container2 [...]] :: name1

   where

   repository1 is the name of the repository
   container1, *container2*, etc are the names of containers
   name1 is the name of ***contained1***

2. The ***absolute_name*** is defined as a ***readonly attribute*** of a ***Contained*** object.

Reference

CORBA, 6.5.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::containing_repository - get repository of contained object

Synopsis - C

```
CORBA_Repository    CORBA_Contained__get_containing_repository
                        ( CORBA_Contained          contained1,
                        CORBA_Environment          * ev1 ) ;
```

Arguments

contained1	*(in)* the contained object
ev1	*(in/out)* the CORBA Environment

Returns

the repository object which contains ***contained1***

Exceptions

(standard)

Description

Returns the repository object which contains the ***Contained*** object.

Notes

1. The ***containing_repository*** can also be obtained by recursively following the object's ***defined_in*** attribute.

2. The ***containing_repository*** is defined as a ***readonly attribute*** of a ***Contained*** object.

Reference

CORBA, 6.5.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::def_kind - get definition kind of a contained object

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::defined_in - get container of contained object

Synopsis - C

```
CORBA_Container     CORBA_Contained__get_defined_in
                        ( CORBA_Contained          contained1,
                        CORBA_Environment          * ev1 ) ;
```

Arguments

contained1	*(in)* the contained object
ev1	*(in/out)* the CORBA Environment

Returns

the object which contains ***contained1***

Exceptions

(standard)

Description

Returns the ***Container*** or ***InterfaceDef*** object which contains the ***Contained*** object.

Notes

1. If the ***Contained*** object is defined within the scope of a ***Container***, then the ***defined_in*** attribute reflects the smallest enclosing scope.

2. If the ***Contained*** object inherits from another interface, then the ***defined_in*** attribute reflects the ***InterfaceDef*** from which the ***Contained*** object is inherited.

3. The ***contained_in*** object is defined as a ***readonly attribute*** of a ***Contained*** object.

Reference

CORBA, 6.5.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::describe - describe contained object

Synopsis - C

```
typedef struct
        {
        CORBA_DefinitionKind        kind ;
        CORBA_any                   value ;
        }
        CORBA_Contained_Description ;

CORBA_Contained_Description     CORBA_Contained_describe
                                        ( CORBA_Contained       contained1,
                                        CORBA_Environment       * ev1 ) ;
```

Arguments

contained1	*(in)* the contained object
ev1	*(in/out)* the CORBA Environment

Returns

a description of the contained object

Exceptions

(standard)

Description

Returns a description of the contained object. The ***value*** within the description is a data structure defined for each ***kind*** of contained object.

Notes

1. Refer to the ***describe*** operation of each interface type (***Attribute***, ***Operation***, etc) for a description of the data structure returned in ***value***.

Reference

CORBA, 6.5.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::destroy - destroy a contained object

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::id - global identifier of a contained object

Synopsis - C

```
typedef CORBA_string    CORBA_RepositoryId ;

CORBA_RepositoryId      CORBA_Contained__get_id
                            ( CORBA_Contained       contained1,
                            CORBA_Environment       * ev1 ) ;
void                    CORBA_Contained__set_id
                            ( CORBA_Contained       contained1,
                            CORBA_RepositoryId      id1,
                            CORBA_Environment       * ev1 ) ;
```

Arguments

contained1	*(in)* the contained object
id1	*(in)* new repository identifier of the contained object
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_Contained__get_id(): the repository identifier of ***contained1***
CORBA_Contained__set_id(): *(void)*

Exceptions

(standard)

Description

Returns the unique identifier of ***contained1*** within its repository.

Notes

1. The ***id*** is the unique, global name of a contained object within its repository.
2. An attempt to change the value of ***id*** to a non-unique value within the repository will cause an exception.
3. The ***id*** is defined as an ***attribute*** of a ***Contained*** object.

Reference

CORBA, 6.5.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::move - move contained object to new container

Synopsis - C

```
void        CORBA_Contained_move
                ( CORBA_Contained       contained1,
                  CORBA_Container       container1,
                  CORBA_Identifier      name1,
                  CORBA_VersionSpec     version1,
                  CORBA_Environment     * ev1 ) ;
```

Arguments

contained1	*(in)* the contained object
container1	*(in)* the new container
name1	*(in)* the new name
version1	*(in)* the new version
ev1	*(in/out)* the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Moves ***contained1*** to a new container within the same repository.

Notes

1. The new container must be capable of containing the object's type.

Reference

CORBA, 6.5.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::name - local identifier of a contained object

Synopsis - C

```
typedef CORBA_string    CORBA_Identifier ;

CORBA_Identifier        CORBA_Contained__get_name
                            ( CORBA_Contained       contained1,
                            CORBA_Environment       * ev1 ) ;
void                    CORBA_Contained__set_name
                            ( CORBA_Contained       contained1,
                            CORBA_Identifier        name1,
                            CORBA_Environment       * ev1 ) ;
```

Arguments

contained1	*(in)* the contained object
id1	*(in)* new local identifier of the contained object
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_Contained__get_name(): the local identifier of ***contained1***
CORBA_Contained__set_name(): *(void)*

Exceptions

(standard)

Description

Returns the unique identifier of ***contained1*** within its smallest containing scope.

Notes

1. The ***id*** is the name of a contained object within its container.
2. The ***name*** is defined as an ***attribute*** of a ***Contained*** object.

Reference

CORBA, 6.5.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Contained::version - get version of a contained object

Synopsis - C

```
typedef CORBA_string    CORBA_VersionSpec ;

CORBA_VersionSpec       CORBA_Contained__get_version
                            ( CORBA_Contained       contained1,
                            CORBA_Environment       * ev1 ) ;
void                    CORBA_Contained__set_version
                            ( CORBA_Contained       contained1,
                            CORBA_VersionSpec       version1,
                            CORBA_Environment       * ev1 ) ;
```

Arguments

contained1	*(in)* the contained object
id1	*(in)* new local version of the contained object
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_Contained__get_version(): the version of ***contained1***
CORBA_Contained__set_version(): *(void)*

Exceptions

(standard)

Description

Returns the version of ***contained1***.

Notes

1. The ***version*** may be used to distinguish among multiple objects with the same name.
2. The ***version*** is defined as an ***attribute*** of a ***Contained*** object.

Reference

CORBA, 6.5.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::contents - get list of contained or inherited objects

Synopsis - C

```
typedef struct
        {
        CORBA_unsigned_long         _maximum ;
        CORBA_unsigned_long         _length ;
        CORBA_Contained             * _buffer ;
        }
        CORBA_ContainedSeq ;

CORBA_ContainedSeq   * CORBA_Container_contents
                            ( CORBA_Container           container1,
                            CORBA_DefinitionKind        limit_type1,
                            CORBA_boolean               exclude_inherited1,
                            CORBA_Environment           * ev1 ) ;
```

Arguments

container1	*(in)* the container whose contents are to be searched
limit_type1	*(in)* restriction to kinds of objects returned if = ***dk_all***, then all interface types are returned if = to a specific interface type, then only objects of that type are returned
exclude_inherited1	*(in)* specifies whether inherited objects (if any) are returned if = ***CORBA_TRUE***, then inherited objects are not returned if = ***CORBA_FALSE***, then all contained objects are returned
ev1	*(in/out)* the CORBA Environment

Returns

the contents of the container

Exceptions

(standard)

Description

Returns a list of the objects within a container. Only those object directly contained or defined are returned.

Notes

(none)

Reference

CORBA, 6.5.4.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::create_alias - create an alias definition

Synopsis - C

```
CORBA_AliasDef      CORBA_Container_create_alias
                        ( CORBA_Container       container1,
                        CORBA_RepositoryId      id1,
                        CORBA_Identifier        name1,
                        CORBA_VersionSpec       version1,
                        CORBA_IDLType           original_type1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

container1	*(in)* the container within which the new alias is to be created
id1	*(in)* the new alias's globally-unique repository identifier
name1	*(in)* the new alias's name within the container
version1	*(in)* the new alias's version information
original_type1	*(in)* the new alias's original type
ev1	*(in/out)* the CORBA Environment

Returns

the new alias definition object

Exceptions

(standard)

Description

Creates a new alias definition within a container.

Notes

(none)

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::create_constant - create a constant definition

Synopsis - C

```
CORBA_ConstantDef    CORBA_Container_create_constant
                         ( CORBA_Container          container1,
                         CORBA_RepositoryId         id1,
                         CORBA_Identifier           name1,
                         CORBA_VersionSpec          version1,
                         CORBA_IDLType              type1,
                         CORBA_any                  * value1,
                         CORBA_Environment          * ev1 ) ;
```

Arguments

container1	*(in)* the container within which the new constant is to be created
id1	*(in)* the new constant's globally-unique repository identifier
name1	*(in)* the new constant's name within the container
version1	*(in)* the new constant's version information
type1	*(in)* the new constant's type
value1	*(in)* the new constant's value
ev1	*(in/out)* the CORBA Environment

Returns

the new constant definition object

Exceptions

(standard)

Description

Creates a new constant definition within a container.

Notes

(none)

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::create_enum - create an enumeration definition

Synopsis - C

```
CORBA_EnumDef     CORBA_Container_create_enum
                        ( CORBA_Container       container1,
                        CORBA_RepositoryId      id1,
                        CORBA_Identifier        name1,
                        CORBA_VersionSpec       version1,
                        CORBA_EnumMemberSeq     members1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

container1	*(in)* the container within which the new enumeration is to be created
id1	*(in)* the new enumeration's globally-unique repository identifier
name1	*(in)* the new enumeration's name within the container
version1	*(in)* the new enumeration's version information
members1	*(in)* the new enumeration's members
ev1	*(in/out)* the CORBA Environment

Returns

the new enumeration definition object

Exceptions

(standard)

Description

Creates a new enumeration definition within a container.

Notes

(none)

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::create_interface - create an interface definition

Synopsis - C

```
CORBA_InterfaceDef    CORBA_Container_create_interface
                          ( CORBA_Container          container1,
                          CORBA_RepositoryId         id1,
                          CORBA_Identifier           name1,
                          CORBA_VersionSpec          version1,
                          CORBA_InterfaceDefSeq      * base_interfaces1,
                          CORBA_Environment          * ev1 ) ;
```

Arguments

container1	*(in)* the container within which the new interface is to be created
id1	*(in)* the new interface's globally-unique repository identifier
name1	*(in)* the new interface's name within the container
version1	*(in)* the new interface's version information
base_interfaces1	*(in)* the interfaces from which the new interface inherits
ev1	*(in/out)* the CORBA Environment

Returns

the new interface definition object

Exceptions

(standard)

Description

Creates a new interface definition within a container.

Notes

(none)

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::create_module - create a module definition

Synopsis - C

```
CORBA_ModuleDef     CORBA_Container_create_module
                        ( CORBA_Container       container1,
                        CORBA_RepositoryId      id1,
                        CORBA_Identifier        name1,
                        CORBA_VersionSpec       version1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

container1	*(in)* the container within which the new module is to be created
id1	*(in)* the new module's globally-unique repository identifier
name1	*(in)* the new module's name within the container
version1	*(in)* the new module's version information
ev1	*(in/out)* the CORBA Environment

Returns

the new module definition object

Exceptions

(standard)

Description

Creates a new module definition within a container.

Notes

(none)

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::create_struct - create a structure definition

Synopsis - C

```
CORBA_StructDef     CORBA_Container_create_struct
                        ( CORBA_Container          container1,
                        CORBA_RepositoryId         id1,
                        CORBA_Identifier           name1,
                        CORBA_VersionSpec          version1,
                        CORBA_StructMemberSeq      members1,
                        CORBA_Environment          * ev1 ) ;
```

Arguments

container1	*(in)* the container within which the new structure is to be created
id1	*(in)* the new structure's globally-unique repository identifier
name1	*(in)* the new structure's name within the container
version1	*(in)* the new structure's version information
members1	*(in)* the new structure's members
ev1	*(in/out)* the CORBA Environment

Returns

the new structure definition object

Exceptions

(standard)

Description

Creates a new structure definition within a container.

Notes

(none)

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::create_union - create a union definition

Synopsis - C

```
CORBA_UnionDef      CORBA_Container_create_union
                        ( CORBA_Container          container1,
                        CORBA_RepositoryId         id1,
                        CORBA_Identifier           name1,
                        CORBA_VersionSpec          version1,
                        CORBA_IDLType              discriminator_type1,
                        CORBA_UnionMemberSeq       members1,
                        CORBA_Environment          * ev1 ) ;
```

Arguments

container1	*(in)* the container within which the new union is to be created
id1	*(in)* the new union's globally-unique repository identifier
name1	*(in)* the new union's name within the container
version1	*(in)* the new union's version information
discriminator_type1	*(in)* the new union's discriminator type
members1	*(in)* the new union's members
ev1	*(in/out)* the CORBA Environment

Returns

the new union definition object

Exceptions

(standard)

Description

Creates a new union definition within a container.

Notes

(none)

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::def_kind - get definition kind of a contained object

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::describe_contents - return description of contents

Synopsis - C

```
typedef struct
        {
        CORBA_Contained             contained_object ;
        CORBA_DefinitionKind        kind ;
        CORBA_any                   value ;
        }
        CORBA_Container_Description ;

typedef struct
        {
        CORBA_unsigned_long         _maximum ;
        CORBA_unsigned_long         _length ;
        CORBA_ContainerDescription  * _buffer ;
        }
        CORBA_Container_DescriptionSeq ;

CORBA_Container_DescriptionSeq
                        * CORBA_Container_describe_contents
                                ( CORBA_Container           container1,
                                CORBA_DefinitionKind        limit_type1,
                                CORBA_boolean               exclude_inherited1,
                                CORBA_long                  max_returned_objs1,
                                CORBA_Environment           * ev1 ) ;
```

Arguments

container1	*(in)* the container whose contents are to be described
limit_type1	*(in)* restriction to kinds of objects returned if = ***dk_all***, then all interface types are returned if = to a specific interface type, then only objects of that type are returned
exclude_inherited1	*(in)* specifies whether inherited objects (if any) are returned if = ***CORBA_TRUE***, then inherited objects are not returned if = ***CORBA_FALSE***, then all contained objects are returned
max_returned_objs1	*(in)* maximum number of objects to be returned ; if = *-1*, then all contained objects are returned
ev1	*(in/out)* the CORBA Environment

Returns

descriptions of the objects within the container

Exceptions

(standard)

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Description

Describes the objects within a container.

Notes

1. The ***value*** element is the same as would be returned by the object's ***describe*** operation.

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::destroy - destroy a container and its contents

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::lookup - locate a definition relative to a container

Synopsis - C

```
CORBA_Contained     CORBA_Container_lookup
                        ( CORBA_Container       container1,
                        CORBA_ScopedName        search_name1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

container1	*(in)* the container whose contents are to be searched
search_name1	*(in)* the name of the object to be found
ev1	*(in/out)* the CORBA Environment

Returns

the object from within the container

Exceptions

(standard)

Description

Searches for an object by name within a container.

Notes

1. If no object is found, a nil object reference is returned.
2. If ***search_name*** begins with *"::"*, then the search begins at the enclosing repository.

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Container::lookup_name - locate objects by name

Synopsis - C

```
typedef struct
        {
        CORBA_unsigned_long         _maximum ;
        CORBA_unsigned_long         _length ;
        CORBA_Contained             * _buffer ;
        }
        CORBA_ContainedSeq ;

CORBA_ContainedSeq   * CORBA_Container_lookup_name
                            ( CORBA_Container           container1,
                            CORBA_ScopedName            search_name1,
                            CORBA_long                  levels_to_search1,
                            CORBA_DefinitionKind        limit_type1,
                            CORBA_boolean               exclude_inherited1,
                            CORBA_Environment           * ev1 ) ;
```

Arguments

container1	*(in)* the container whose contents are to be searched
search_name1	*(in)* the name of the object to be found
levels_to_search1	*(in)* the maximum number of levels to search if = *-1*, then the current object and all contained objects are searched if = *1*, then only the current object is searched
limit_type1	*(in)* restriction to kinds of objects returned if = ***dk_all***, then all interface types are returned if = to a specific interface type, then only objects of that type are returned
exclude_inherited1	*(in)* specifies whether inherited objects (if any) are returned if = ***CORBA_TRUE***, then inherited objects are not returned if = ***CORBA_FALSE***, then all contained objects are returned
ev1	*(in/out)* the CORBA Environment

Returns

the objects from within the container which meet the search criteria

Exceptions

(standard)

Description

Searches for objects by name within a container.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Notes

1. If ***search_name*** begins with *"::"*, then the search begins at the enclosing repository.

Reference

CORBA, 6.5.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Context::create_child - create a child context object

Synopsis - C

```
CORBA_Status   CORBA_Context_create_child ( CORBA_Context ctx11,
                                            CORBA_Identifier name1, CORBA_Context * ctx2,
                                            CORBA_Environment * ev1 ) ;
```

Arguments

ctx1	(in) the context object for which a child is to be created
name1	(in) the name of the new child context object
ctx2	(out) the new child context object
ev1	(in/out) the CORBA Environment

Returns

the operation status

Exceptions

(standard)

Description

Creates a child context object.

The child object is chained to its parent context. Searches on the child context will look into the parent context (and on up the tree) to find property values.

Notes

(none)

Reference

CORBA, 4.6.6

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Context::delete - delete the indicated context object

Synopsis - C

```
CORBA_Status   CORBA_Context_delete ( CORBA_Context ctx1,
                                      CORBA_Flags flags1,
                                      CORBA_Environment * ev1 ) ;
```

Arguments

ctx1	(in) the context object to be deleted
flags1	(in) option flags
ev1	(in/out) the CORBA Environment

Returns

the operation status

Exceptions

(standard)

Description

Deletes the specified context object.

The only option flag currently defined is ***CORBA::CTX_DELETE_DESCENDENTS***, which causes the deletion of all descendent context objects.

Notes

1. An exception is returned if there are one or more child context objects and the was ***CORBA::CTX_DELETE_DESCENDENTS flag*** was not set.

Reference

CORBA, 4.6.7.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Context::delete_values - delect the specified property values

Synopsis - C

```
CORBA_Status   CORBA_Context_delete_values ( CORBA_Context ctx1,
                                             CORBA_Identifier name1,
                                             CORBA_Environment * ev1 ) ;
```

Arguments

ctx1	(in) the context object from which the value or values are to be deleted
name1	(in) the name of the property values to be deleted
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Deletes one or more property values from a context object. The search is limited to the current object (not including parent contexts).

The property name ***name1*** supports the optional use of a single trailing wildcard character (asterisk) for multiple properties.

Notes

(none)

Reference

CORBA, 4.6.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Context::get_values - retrieve the specified context property values

Synopsis - C

```
CORBA_Status   CORBA_Context_get_values ( CORBA_Context ctx1,
                                          CORBA_Identifier scope1, CORBA_Flags flags1,
                                          CORBA_Identifer name1, CORBA_NVList * list1,
                                          CORBA_Environment * ev1 ) ;
```

Arguments

ctx1	(in) the context object to be searched
scope1	(in) the level at which to initiate the search (for example, ***_USER***, ***_SYSTEM***, etc)
flags1	(in) operation flags
name1	(in) name of property
list1	(out) list of properties and values
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Retrieves one or more property values from a context object. A single trailing wildcard character (asterisk) may be used in ***name1*** to specify multiple values.

The only value currently defined for ***flags1*** is ***CORBA::CTX_RESTRICT_SCOPE***, which causes the search to be restricted to the current object or scope (without chaining).

Notes

(none)

Reference

CORBA, 4.6.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Context::set_one_value - set a single context object property

Synopsis - C

```
CORBA_status   CORBA_Context_set_one_value ( CORBA_Context ctx1,
                                             CORBA_Identifier name1, CORBA_string value1,
                                             CORBA_Environment * ev1 ) ;
```

Arguments

ctx1	(in) the context object whose property is to be set
name1	(in) the name of the property
value1	(in) the value of the property
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Sets a single property value in a context object.

Notes

(none)

Reference

CORBA, 4.6.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Context::set_values - set one or more property values in a context object

Synopsis - C

```
CORBA_Status   CORBA_Context_set_values ( CORBA_Context ctx1,
                                          CORBA_NVList list1,
                                          CORBA_Environment * ev1 ) ;
```

Arguments

ctx1	(in) the context object whose property values are to be set
list1	(in) a list of property names and values
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Sets one or more property values in a context object.

Notes

1. In ***list1***, the flags field must set to zero and type ***TypeCode*** fields must be ***TC_string***.

Reference

CORBA, 4.6.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::DomainManager::get_domain_policy - get policy for objects in domain

Synopsis - C

```
CORBA_Policy        CORBA_DomainManager_get_domain_policy
                        ( CORBA_DomainManager domain_manager1,
                        CORBA_PolicyType policy_type1,
                        CORBA_Environment * env1 ) ;
```

Arguments

domain_manager1	*(in)* the domain manager
policy_type1	*(in)* the type of policy object sought
ev1	*(in/out)* the CORBA Environment

Returns

the policy object of the specified type for the objects in the domain

Exceptions

(standard)

Description

Determines the policy of the specified type for objects in the domain.

Notes

1. Some policies for domains are defined by the security service.

Reference

CORBA 2.1, 5.9.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::EnumDef::absolute_name - get scoped name of enumeration definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::absolute_name

CORBA::EnumDef::containing_repository - get repository of enumeration definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::containing_repository

CORBA::EnumDef::def_kind - get definition kind of a enumeration definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::EnumDef::defined_in - get container of enumeration definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::EnumDef::describe - describe enumeration definition

Inherited from

CORBA::TypedefDef

Refer to

CORBA::TypedefDef::describe

CORBA::EnumDef::destroy - destroy an enumeration definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::EnumDef::id - global identifier of an enumeration definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::EnumDef::members - members of a enumeration

Synopsis - C

```
typedef
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_Identifier                * _buffer ;
        }
        CORBA_EnumMemberSeq ;

CORBA_EnumMemberSeq     * CORBA_EnumDef__get_members
                                ( CORBA_EnumDef             enumdef1,
                                CORBA_Environment           * ev1 ) ;

void                    CORBA_EnumDef__set_members
                                ( CORBA_EnumDef             enumdef1,
                                CORBA_EnumMemberSeq         * members1,
                                CORBA_Environment           * ev1 ) ;
```

Arguments

enumdef1	*(in)* the enumeration definition object
members1	*(in)* new members descriptions
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_EnumDef__get_members(): member description sequence
CORBA_EnumDef__set_members(): *(void)*

Exceptions

(standard exceptions)

Description

Returns or sets the members of an enumeration definition.

Notes

1. Each identifier of the sequence must be unique within the sequence.

Reference

CORBA, 6.5.12.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::EnumDef::move - move enumeration definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::EnumDef::name - local identifier of an enumeration definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

CORBA::EnumDef::type - type of an enumeration definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

CORBA::EnumDef::version - get version of an enumeration definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_exception_as_any ()- return value of current exception

Synopsis - C

```
CORBA_any       * CORBA_exception_as_any ( CORBA_Environment * ev1 ) ;
```

Arguments

ev1 *(in/out)* the CORBA Environment

Returns

a pointer to a ***CORBA_any*** containing the exception value

Exceptions

(none)

Description

Returns the value of the current exception.

Notes

1. If no exception condition exists, then ***NULL*** is returned.

2. Ownership of the storage associated with the return parameter remains with the system, and does not transfer to the caller. The pointer remains valid until either (a) ***CORBA_exception_free()*** is called, or (b) any other CORBA routine which might create an exception condition is called.

Reference

CORBA, 14.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_exception_free ()- clear exception and release associated storage

Synopsis - C

```
void    CORBA_exception_free ( CORBA_Environment * ev1 ) ;
```

Arguments

ev1	*(in/out)* the CORBA Environment

Returns

(void)

Exceptions

(none)

Description

Clears exception condition, releasing associated storage.

Notes

1. It is not an error to call this procedure even if no exception condition exists.

Reference

CORBA, 14.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_exception_id ()- return identification of current exception

Synopsis - C

```
CORBA_char      * CORBA_exception_id ( CORBA_Environment * ev1 ) ;
```

Arguments

ev1	*(in/out)* the CORBA Environment

Returns

a pointer to the character string identifying the current exception

Exceptions

(none)

Description

Returns identification of the current exception. The returned string is the repository ID string of the the exception.

Notes

1. If no exception condition exists, then ***NULL*** is returned.
2. and does not transfer to the caller. The pointer remains valid until either (a) ***CORBA_exception_free()*** is called, or (b) any other CORBA routine which might create an exception condition is called.

Reference

CORBA, 14.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_exception_set ()- raise an exception condition

Synopsis - C

```
void        CORBA_exception_set ( CORBA_Environment * env1,
                                  CORBA_exception_type      major1,
                                  CORBA_char                * repository_id1,
                                  void                      * value1 ) ;
```

Arguments

ev1	*(in/out)* the CORBA Environment
major1	*(in)* one of ***CORBA_NO_EXCEPTION***, ***CORBA_USER_EXCEPTION***, or ***CORBA_SYSTEM_EXCEPTION***
repository_id1	*(in)* repository identifier of exception
value1	*(in)* value of the exception structure

Returns

(void)

Exceptions

(none)

Description

Sets the current exception value, thereby raising an exception condition.

Notes

1. If ***major1 = CORBA_NO_EXCEPTION***, then ***repository_id1*** and ***value1*** must be ***NULL***.

2. Ownership of the storage for the repository ID and value are transferred from the caller to the system by this routine. Such storage should be dynamically allocated by the caller before passing to this routine. (The appropriate allocation function ***T__alloc()*** can be used for the value, where ***T*** is the exception type. The repository ID can be allocated using ***CORBA_string_alloc()***.)

3. Exceptions are freed in advance by the object adaptor before calling method implementations, so this routine does not need to be called in a method implementation if no exception is to be raised.

Reference

CORBA, 14.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_exception_value ()- return exception data structure

Synopsis - C

```
void    * CORBA_exception_value ( CORBA_Environment * ev1 ) ;
```

Arguments

ev1	*(in/out)* the CORBA Environment

Returns

a pointer to the data structure corresponding to the current exception

Exceptions

(none)

Description

Returns data structure for the current exception.

Notes

1. If no exception condition exists, then ***NULL*** is returned.

2. Ownership of the storage associated with the return parameter remains with the system, and does not transfer to the caller. The pointer remains valid until either (a) ***CORBA_exception_free()*** is called, or (b) any other CORBA routine which might create an exception condition is called.

Reference

CORBA, 14.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ExceptionDef::absolute_name - get scoped name of exception definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::absolute_name

CORBA::ExceptionDef::containing_repository - get repository of exception definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::containing_repository

CORBA::ExceptionDef::def_kind - get definition kind of an exception definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::ExceptionDef::defined_in - get container of exception definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ExceptionDef::describe - describe exception definition

Synopsis - C

```
typedef struct
        {
        CORBA_DefinitionKind            kind ;
        CORBA_any                       value ;
        }
        CORBA_Contained_Description ;

typedef struct
        {
        CORBA_Identifier                name ;
        CORBA_RepositoryId              id ;
        CORBA_RepositoryId              defined_in ;
        CORBA_VersionSpec               version ;
        CORBA_TypeCode                  type ;
        }
        CORBA_ExceptionDescription ;

CORBA_Contained_Description     CORBA_ExceptionDef_describe
                                        ( CORBA_ExceptionDef    exceptiondef1,
                                        CORBA_Environment       * ev1 ) ;
```

Arguments

exception1	*(in)* the module definition object
ev1	*(in/out)* the CORBA Environment

Returns

a description of the exception definition object

Exceptions

(standard)

Description

Returns a description of an exception definition object. The ***value*** within the description is the ***CORBA::ExceptionDescription*** data structure.

Notes

1. Inherited from ***CORBA::Contained***. Other interfaces which inherit from ***CORBA::Contained*** may return different structures for the ***value*** element of ***CORBA::Contained::Description***.

Reference

CORBA, 6.5.19.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ExceptionDef::destroy - destroy an exception definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::ExceptionDef::id - global identifier of an exception definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ExceptionDef::members - members of an exception definition

Synopsis - C

```
typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_TypeCode              type ;
        CORBA_IDLType               type_def ;
        }
        CORBA_StructMember ;

typedef
        {
        unsigned long               _maximum ;
        unsigned long               _length ;
        CORBA_StructMember          * _buffer ;
        }
        CORBA_StructMemberSeq ;

CORBA_StructMemberSeq       * CORBA_ExceptionDef__get_members
                                ( CORBA_ExceptionDef        exceptiondef1,
                                CORBA_Environment           * ev1 ) ;

void                        CORBA_ExceptionDef__set_members
                                ( CORBA_ExceptionDef        exceptiondef1,
                                CORBA_StructMemberSeq       * members1,
                                CORBA_Environment           * ev1 ) ;
```

Arguments

exceptiondef1	*(in)* the structure definition object
members1	*(in)* new members descriptions
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_ExceptionDef__get_members(): member description sequence
CORBA_ExceptionDef__set_members(): *(void)*

Exceptions

(standard exceptions)

Description

Returns or sets the members of an exception definition.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Notes

1. When setting the ***members*** attribute (using ***CORBA_StructDef__set_members()***), the ***type*** member of each ***CORBA::StructMember*** structure should be set to ***TC_void***. The value of the ***type_def*** member will automatically cause that of the ***type*** member value to be updated correctly.

Reference

CORBA, 6.5.19.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ExceptionDef::move - move exception definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::ExceptionDef::name - local identifier of an exception definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ExceptionDef::type - type code of exception definition

Synopsis - C

```
CORBA_TypeCode      CORBA_ExceptionDef__get_type
                        ( CORBA_ExceptionDef        exceptiondef1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

exceptiondef1	*(in)* the exception definition object
ev1	*(in/out)* the CORBA Environment

Returns

the typecode of the exception definition object

Exceptions

(standard)

Description

Returns the typecode for an exception definition object.

Notes

(none)

Reference

CORBA, 6.5.19.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ExceptionDef::version - get version of an exception definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

CORBA::IDLType::def_kind - get definition kind of a IDL type

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

CORBA::IDLType::destroy - destroy an IDL type

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::IDLType::type - type of an IDL type

Synopsis - C

```
CORBA_TypeCode      CORBA_IDLType__get_type
                        ( CORBA_IDLType         idltype1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

idltype1	*(in)* the IDL type object
ev1	*(in/out)* the CORBA Environment

Returns

the typecode of the IDL type object

Exceptions

(standard)

Description

Returns the typecode for an IDL type object.

Notes

(none)

Reference

CORBA, 6.5.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::base_interfaces - base interfaces inherited by an interface

Synopsis - C

```
typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_InterfaceDef              * _buffer ;
        }
        CORBA_InterfaceDefSeq ;

CORBA_InterfaceDefSeq       * CORBA_InterfaceDef__get_base_interfaces
                                    ( CORBA_InterfaceDef            interfacedef1,
                                    CORBA_Environment               * ev1 ) ;

void                    CORBA_InterfaceDef__set_base_interfaces
                                ( CORBA_InterfaceDef            interfacedef1,
                                CORBA_InterfaceDefSeq           * base_interfaces1,
                                CORBA_Environment               * ev1 ) ;
```

Arguments

interfacedef1	*(in)* the interface definition object
base_interfaces1	*(in)* the new base interfaces
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_InterfaceDef__get_base_interfaces(): the interfaces inherited by the interface
CORBA_InterfaceDef__set_base_interfaces(): *(void)*

Exceptions

(standard)

Description

Return or set the interfaces inherited by an interface.

Notes

(none)

Reference

CORBA, 6.5.22.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::contents - get list of contained or inherited objects

Inherited from

CORBA::Container

Refer to

CORBA::Container::contents

CORBA::InterfaceDef::create_alias - create an alias definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_alias

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::create_attribute - create an attribute definition

Synopsis - C

```
CORBA_AttributeDef    CORBA_InterfaceDef_create_attribute
                          ( CORBA_InterfaceDef      interfacedef1,
                          CORBA_RepositoryId        id1,
                          CORBA_Identifier          name1,
                          CORBA_VersionSpec         version1,
                          CORBA_IDLType             type1,
                          CORBA_AttributeMode       mode1,
                          CORBA_Environment         * ev1 ) ;
```

Arguments

interfacedef1	*(in)* the interface definition object to contain the new attribute
id1	*(in)* the globally-unique identifier of the ***AttributeDef*** within the repository
name1	*(in)* the name of the ***AttributeDef*** within the ***InterfaceDef***
version1	*(in)* the version of the ***AttributeDef***
type1	*(in)* the IDL type of the ***AttributeDef***
mode1	*(in)* the attribute mode (either ***CORBA_ATTR_NORMAL*** or ***CORBA_ATTR_READONLY***)
ev1	*(in/out)* the CORBA Environment

Returns

the new attribute definition

Exceptions

(standard)

Description

Create a new attribute definition within an interface.

Notes

(none)

Reference

CORBA, 6.5.22.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::create_constant - create a constant definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_constant

CORBA::InterfaceDef::create_enum - create an enumeration definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_enum

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::create_operation - create an operation definition

Synopsis - C

```
CORBA_OperationDef   CORBA_InterfaceDef_create_operation
                         ( CORBA_InterfaceDef          interfacedef1,
                         CORBA_RepositoryId            id1,
                         CORBA_Identifier              name1,
                         CORBA_VersionSpec             version1,
                         CORBA_IDLType                 result1,
                         CORBA_OperationMode           mode1,
                         CORBA_ParDescriptionSeq       * params1,
                         CORBA_ExceptionDefSeq         * exceptions1,
                         CORBA_ContextIdSeq            * contexts1,
                         CORBA_Environment             * ev1 ) ;
```

Arguments

interfacedef1	*(in)* the interface definition object to contain the new operation
id1	*(in)* the globally-unique identifier of the ***OperationDef*** within the repository
name1	*(in)* the name of the ***OperationDef*** within the ***InterfaceDef***
version1	*(in)* the version of the ***OperationDef***
result1	*(in)* the IDL type of the result
mode1	*(in)* the operation mode (either ***CORBA_OP_NORMAL*** or ***CORBA_OP_ONEWAY***)
params1	*(in)* the parameters of the new OperationDef
exceptions1	*(in)* the exceptions of the new OperationDef
contexts1	*(in)* the contexts of the new OperationDef
ev1	*(in/out)* the CORBA Environment

Returns

the new operation definition

Exceptions

(standard)

Description

Create a new operation definition within an interface.

Notes

(none)

Reference

CORBA, 6.5.22.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::create_struct - create a structure definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_struct

CORBA::InterfaceDef::create_union - create a union definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_union

CORBA::InterfaceDef::def_kind - get definition kind of an interface definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::InterfaceDef::defined_in - get container of interface definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::describe - describe interface definition

Synopsis - C

```
typedef struct
        {
        CORBA_DefinitionKind        kind ;
        CORBA_any                   value ;
        }
        CORBA_Contained_Description ;

typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_RepositoryId          id ;
        CORBA_RepositoryId          defined_in ;
        CORBA_VersionSpec           version ;
        CORBA_RepositoryIdSeq       base_interfaces ;
        }
        CORBA_InterfaceDescription ;

CORBA_Contained_Description     CORBA_InterfaceDef_describe
                                        ( CORBA_InterfaceDef    interfacedef1,
                                        CORBA_Environment       * ev1 ) ;
```

Arguments

interfacedef1	*(in)* the interface definition object
ev1	*(in/out)* the CORBA Environment

Returns

a description of the interface definition object

Exceptions

(standard)

Description

Returns a description of an interface definition object. The ***value*** within the description is the ***CORBA::InterfaceDescription*** data structure.

Notes

1. Inherited from ***CORBA::Contained***. Other interfaces which inherit from ***CORBA::Contained*** may return different structures for the ***value*** element of ***CORBA::Contained::Description***.

2. See also ***CORBA::InterfaceDef::describe_interface***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

CORBA, 6.5.22.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::describe_contents - return description of contents

Inherited from

CORBA::Container

Refer to

CORBA::Container::describe_contents

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::describe_interface - return full interface description

Synopsis - C

```
typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_RepositoryId          id ;
        CORBA_RepositoryId          defined_in ;
        CORBA_VersionSpec           version ;
        CORBA_OpDescriptionSeq      operations ;
        CORBA_AttrDescriptionSeq    attributes ;
        CORBA_RepositoryIdSeq       base_interfaces ;
        CORBA_TypeCode              type ;
        }
        CORBA_InterfaceDef_FullInterfaceDescription ;

CORBA_InterfaceDef_FullInterfaceDescription
                                CORBA_InterfaceDef_describe
                                        ( CORBA_InterfaceDef    interfacedef1,
                                        CORBA_Environment       * ev1 ) ;
```

Arguments

interfacedef1	*(in)* the interface definition object
ev1	*(in/out)* the CORBA Environment

Returns

a description of the interface definition object

Exceptions

(standard)

Description

Returns a description of an interface definition object.

Notes

1. See also ***CORBA::InterfaceDef::describe***.

Reference

CORBA, 6.5.22.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::destroy - destroy an interface definition and its contents

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::InterfaceDef::id - global identifier of an interface definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::is_a - test for interface equivalence or inheritance

Synopsis - C

```
CORBA_boolean     * CORBA_InterfaceDef_is_a
                        ( CORBA_InterfaceDef      interfacedef1,
                        CORBA_RepositoryId        interface_id2,
                        CORBA_Environment         * ev1 ) ;
```

Arguments

interfacedef1	*(in)* the interface definition object
interface_id2	*(in)* the repository identifier of the interface to be tested
ev1	*(in/out)* the CORBA Environment

Returns

if ***interfacedef1*** is identical to or inherits (directly or indirectly) from ***interface_id2***, then ***CORBA_TRUE***,

else ***CORBA_FALSE***

Exceptions

(standard)

Description

Determines if the interface object is identical to or inherits from a certain other interface.

Notes

(none)

Reference

CORBA, 6.5.22.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::lookup - locate a definition relative to a container

Inherited from

CORBA::Container

Refer to

CORBA::Container::lookup

CORBA::InterfaceDef::lookup_name - locate an object by name

Inherited from

CORBA::Container

Refer to

CORBA::Container::lookup_name

CORBA::InterfaceDef::move - move interface definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::InterfaceDef::name - local identifier of an interface definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::InterfaceDef::type - type code of an interface definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

CORBA::InterfaceDef::version - get version of an interface definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::IRObject::def_kind - type of interface repository definition

Synopsis - C

```
CORBA_DefinitionKind  CORBA_IRObject__get_def_kind ( CORBA_IRObject irobj1,
                                                     CORBA_Environment * ev1 ) ;
```

Arguments

irobj1	(in) the interface repository object to be examined
ev1	(in/out) the CORBA Environment

Returns

the type of definition represented by the object

Exceptions

(standard)

Description

Obtains the definition kind of an interface repository object.

Notes

(none)

Reference

CORBA, 6.5.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::IRObject::destroy - cause interface repository object to cease existence

Synopsis - C

```
void    CORBA_IRObject_destroy ( CORBA_IRObject irobj1,
                CORBA_Environment * ev1 ) ;
```

Arguments

irobj1	(in) the interface repository object to be destroyed
ev1	(in/out) the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Causes an interface repository object to cease to exist.

If ***irobj1*** is a container, the operation is applied to all of its contents. If ***irobj1*** is contained in some other object, it is removed.

Notes

1. It is an error to apply this operation to a ***CORBA::Repository*** or to a ***CORBA::PrimitiveDef*** object.

Reference

CORBA, 6.5.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ModuleDef::contents - get list of contained or inherited objects

Inherited from

CORBA::Container

Refer to

CORBA::Container::contents

CORBA::ModuleDef::create_alias - create an alias definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_alias

CORBA::ModuleDef::create_constant - create a constant definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_constant

CORBA::ModuleDef::create_enum - create an enumeration definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_enum

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ModuleDef::create_interface - create an interface definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_interface

CORBA::ModuleDef::create_module - create a module definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_module

CORBA::ModuleDef::create_struct - create a structure definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_struct

CORBA::ModuleDef::create_union - create a union definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_union

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ModuleDef::def_kind - get definition kind of a module definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::ModuleDef::defined_in - get container of a module definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ModuleDef::describe - describe module definition

Synopsis - C

```
typedef struct
        {
        CORBA_DefinitionKind        kind ;
        CORBA_any                   value ;
        }
        CORBA_Contained_Description ;

typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_RepositoryId          id ;
        CORBA_RepositoryId          defined_in ;
        CORBA_VersionSpec           version ;
        }
        CORBA_ModuleDescription ;

CORBA_Contained_Description     CORBA_ModuleDef_describe
                                        ( CORBA_ModuleDef       moduledef1,
                                        CORBA_Environment       * ev1 ) ;
```

Arguments

moduledef1	*(in)* the module definition object
ev1	*(in/out)* the CORBA Environment

Returns

a description of the module definition object

Exceptions

(standard)

Description

Returns a description of a module definition object. The ***value*** within the description is the ***CORBA::ModuleDescription*** data structure.

Notes

1. Inherited from ***CORBA::Contained***. Other interfaces which inherit from ***CORBA::Contained*** may return different structures for the ***value*** element of ***CORBA::Contained::Description***.

Reference

CORBA, 6.5.7.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ModuleDef::describe_contents - return description of contents

Inherited from

CORBA::Container

Refer to

CORBA::Container::describe_contents

CORBA::ModuleDef::destroy - destroy a module definition and its contents

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::ModuleDef::id - global identifier of a module definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

CORBA::ModuleDef::lookup - locate a definition relative to a container

Inherited from

CORBA::Container

Refer to

CORBA::Container::lookup

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ModuleDef::lookup_name - locate an object by name

Inherited from

CORBA::Container

Refer to

CORBA::Container::lookup_name

CORBA::ModuleDef::move - move module definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::ModuleDef::name - local identifier of a module definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

CORBA::ModuleDef::version - get version of a module definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::NVList::add_item - add an item to a named value list

Synopsis - C

```
CORBA_Status   CORBA_NVList_add_item ( CORBA_NVList list1,
                                       CORBA_Identifier name1, CORBA_TypeCode type1,
                                       void *value1, CORBA_long length1, CORBA_Flags flags1,
                                       CORBA_Environment * ev1 ) ;
```

Arguments

list1	(in) the named value list to which the item is to be added
name1	(in) name of item
type1	(in) type of item
value1	(in) value of item
length1	(in) length or count of item
flags1	(in) item flags
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Adds a value to a named item list.

flags1 may be the following:

- one of ***CORBA::ARG_IN***, ***CORBA::ARG_OUT***, or ***CORBA::ARG_INOUT***.
- ***CORBA::IN_COPY_VALUE*** will cause the operation to make a copy of the request value (***value1***) instead of assuming ownership of the storage.
- ***CORBA::DEPENDENT_LIST***, when a list structure is added as an item, will indicate that the sublist should be freed when the parent list is freed.

Notes

(none)

Reference

CORBA, 4.4.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::NVList::free - free a named value list and associated storage

Synopsis - C

```
CORBA_Status   CORBA_NVList_free ( CORBA_NVList       list1,
                                   CORBA_Environment * ev1 ) ;
```

Arguments

list1	(in) the list to be freed
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Frees a named value list and associated storage.

Notes

(none)

Reference

CORBA, 4.4.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::NVList::free_memory - free argument-out memory of named value list

Synopsis - C

```
CORBA_Status   CORBA_NVList_free_memory ( CORBA_NVList list1,
                                          CORBA_Environment * ev1 ) ;
```

Arguments

list1	(in) the list whose argument-out memory is to be freed
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Free the dynamically-allocated out-argument list associated with a named value list.

Notes

(none)

Reference

CORBA, 4.4.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::NVList::get_count - return the number of items in a named value list

Synopsis - C

```
CORBA_Status   CORBA_NVList_get_count ( CORBA_NVList list1,
                                        CORBA_long * count1,
                                        CORBA_Environment * ev1 ) ;
```

Arguments

list1	(in) the list whose items are to be counted
count1	(out) the number of items in the list
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Returns the number of items in a named value list.

Notes

(none)

Reference

CORBA, 4.4.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::create_request - create an ORB request

Synopsis - C

```
typedef CORBA_unsigned_long   CORBA_Status ;
typedef CORBA_Object          CORBA_Context ;
typedef CORBA_Object          CORBA_NVList ;
typedef CORBA_Object          CORBA_Request ;
typedef CORBA_unsigned_long   CORBA_Flags ;
typedef CORBA_string          CORBA_Identifier ;

CORBA_Status  CORBA_Object_create_request ( CORBA_Object obj1,
                        CORBA_Context ctx1, CORBA_Identifier operation1,
                        CORBA_NVList list1, CORBA_NamedValue * result1,
                        CORBA_Request *request1, CORBA_Flags flags1,
                        CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the object on which the request is to be made
ctx1	(in) context object for the operation
operation1	(in) name of intended operation on obj1
list1	(in) arguments to the operation
result1	(in/out) operation result
request1	(out) newly-created request
flags1	(in) request flags
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Creates an ORB request.

The actual invocation occurs by calling ***CORBA::Request::invoke*** or by using ***CORBA::Request::send*** with ***CORBA::Request::get_response***.

Currently, only one value is defined for ***flags1***. If the ***CORBA::OUT_LIST_MEMORY*** flag is set, then ***list1*** must also be specified, and out-arguement memory allocated for ***request1*** arguments is associated with ***list1***. In this case, when ***list1*** is freed, then the out-argument memory is also freed.

Notes

(none)

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

CORBA, 4.2.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::duplicate - make a copy of an object reference

Synopsis - C

```
CORBA_Object   CORBA_Object_duplicate ( CORBA_Object obj1,
                                        CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the object reference to be duplicated
ev1	(in/out) the CORBA Environment

Returns

a copy of the reference to ***obj1***

Exceptions

(standard)

Description

Makes a copy of an object reference. Since CORBA_Object is an opaque data type, this is the only portable mechanism for this purpose.

Notes

(none)

Reference

CORBA, 7.2.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::hash - compute a hash-code value for an object reference

Synopsis - C

```
CORBA_unsigned_long  CORBA_Object_hash ( CORBA_Object obj1,
                                         CORBA_unsigned_long maximum1,
                                         CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the object for which the hash value is to be computed
maximum1	(in) the maximum value of the hash code space
ev1	(in/out) the CORBA Environment

Returns

a hash-code value for the object

Exceptions

(standard)

Description

Compute a hash-code value for an object reference. The hash code will not change during the lifetime of the object reference.

Hash codes can be used to eliminate the possibility of two references referring to the same object. However, if two codes are identical, the equivalence of the object is not proven.

Notes

(none)

Reference

CORBA, 7.2.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::get_domain_managers - get domain managers for an object

Synopsis - C

```
CORBA_DomainManagersList    * CORBA_Object_get_domain_managers
                                    ( CORBA_Object obj1,
                                    CORBA_Environment * env1 ) ;
```

Arguments

obj1	*(in)* the object for which the policy is to be determined
env1	*(in/out)* the CORBA Environment

Returns

the domain managers associated with the specified object

Exceptions

(standard)

Description

Returns the immediately-enclosing domain managers for the object.

Notes

1. Because every object is, by default, associated with a domain manager, the returned list always contains at least one item.

Reference

CORBA, 5.2.8.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::get_implementation - determine the implementation of an object

Synopsis - C

```
typedef CORBA_Object   CORBA_ImplementationDef ;

CORBA_ImplementationDef      CORBA_Object_get_implementation
                                    ( CORBA_Object obj1,
                                    CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the object whose implementation is sought
ev1	(in/out) the CORBA Environment

Returns

the implementation of the object

Exceptions

(standard)

Description

Returns the ***ImplementationDef*** used to define the object in ***CORBA::BOA::create***.

Notes

(none)

Reference

CORBA, 7.2.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::get_interface - return the interface of an object

Synopsis - C

```
typedef CORBA_Object   CORBA_InterfaceDef ;

CORBA_InterfaceDef     CORBA_Object_get_interface ( CORBA_Object obj1,
                           CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the object for which the interface is to be determined
ev1	(in/out) the CORBA Environment

Returns

the interface for the object

Exceptions

(standard)

Description

Returns the ***InterfaceDef*** used to define the object in ***CORBA::BOA::create***.

Notes

(none)

Reference

CORBA, 7.2.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::get_policy - determine policy of given type

Synopsis - C

```
CORBA_Policy   CORBA_Object_get_policy ( CORBA_Object obj1,
                                         CORBA_PolicyType policy_type1,
                                         CORBA_Environment * env1 ) ;
```

Arguments

obj1	*(in)* the object for which the policy is to be determined
policy_type1	*(in)* the type of policy
env1	*(in/out)* the CORBA Environment

Returns

the policy object of the specified type

Exceptions

(standard)

Description

Returns the policy object for the specified policy type.

Notes

(none)

Reference

CORBA, 5.2.7.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::is_a - check if object is instance of a type

Synopsis - C

```
CORBA_boolean     CORBA_Object_is_a ( CORBA_Object obj1,
                                      CORBA_string   logical_type_id1,
                                      CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the object whose type is to be checked
logical_type_id1	(in) the shared type identifier (***RepositoryId***) to be checked
ev1	(in/out) the CORBA Environment

Returns

CORBA_TRUE if ***obj1*** is an instance of ***logical_type_id1***, else ***CORBA_FALSE***

Exceptions

(standard)

Description

Used for type checking of objects.

Notes

(none)

Reference

CORBA, 7.2.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::is_equivalent - determine if two object references are equivalent

Synopsis - C

```
CORBA_boolean     CORBA_Object_is_equivalent ( CORBA_Object obj1,
                                               CORBA_Object obj2,
                                               CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the first object reference to be checked
obj2	(in) the second object reference to be checked
ev1	(in/out) the CORBA Environment

Returns

CORBA_TRUE if both object references are equivalent, otherwise ***CORBA_FALSE***.

Exceptions

(standard)

Description

Checks if two object references are equivalent.

If ***CORBA_FALSE*** is returned, there is a possibility that the two references are equivalent, but the system was unable to determine this was so.

Notes

(none)

Reference

CORBA, 7.2.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::is_nil - determine if an object reference is to no object

Synopsis - C

```
CORBA_boolean       CORBA_Object_is_nil ( CORBA_Object obj1,
                                          CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the object reference to be tested
ev1	(in/out) the CORBA Environment

Returns

CORBA_TRUE if the value is ***CORBA_OBJECT_NIL***, else ***CORBA_FALSE***

Exceptions

(standard)

Description

Determines if the reference is to no object.

Notes

(none)

Reference

CORBA, 7.2.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::non_existent - determine if an object exists

Synopsis - C

```
CORBA_boolean     CORBA_Object_non_existent ( CORBA_Object obj1,
                                              CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the object reference to be tested
ev1	(in/out) the CORBA Environment

Returns

CORBA_TRUE if the object reference is known authoritatively not to exist, otherwise ***CORBA_FALSE***

Exceptions

(standard)

Description

Check to see if an object exists.

An object reference might refer to an object which has subsequently been removed: this operation can be used to check for such an occurrence.

Notes

(none)

Reference

CORBA, 7.2.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Object::release - free the storage associated with an object reference

Synopsis - C

```
void    CORBA_Object_release ( CORBA_Object obj1,
                               CORBA_Environment * ev1 ) ;
```

Arguments

obj1	(in) the object reference whose storage is to be released
ev1	(in/out) the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Frees the storage associated with an (opaque) object reference.

Notes

(none)

Reference

CORBA, 7.2.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::absolute_name - get scoped name of operation definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::absolute_name

CORBA::OperationDef::containing_repository - get repository of operation definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::containing_repository

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::contexts - contexts of an operation

Synopsis - C

```
typedef CORBA_Identifier        CORBA_ContextIdentifier ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_ContextIdentifier         * _buffer ;
        }
        CORBA_ContextIdSeq ;

CORBA_ContextIdSeq              * CORBA_OperationDef__get_contexts
                                        ( CORBA_OperationDef            operationdef1,
                                        CORBA_Environment               * ev1 ) ;

void                            CORBA_OperationDef__set_contexts
                                        ( CORBA_OperationDef            operationdef1,
                                        CORBA_ContextIdSeq              * contexts1,
                                        CORBA_Environment               * ev1 ) ;
```

Arguments

operationdef1	*(in)* the operation definition object
contexts1	*(in)* the new context identifiers
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_OperationDef__get_contexts(): the context identifiers of the operation
CORBA_OperationDef__set_contexts(): *(void)*

Exceptions

(standard)

Description

Return or set the context identifiers of an operation.

Notes

(none)

Reference

CORBA, 6.5.21.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::def_kind - get definition kind of a operation definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::OperationDef::defined_in - get container of operation definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::describe - describe operation definition

Synopsis - C

```
typedef struct
        {
        CORBA_DefinitionKind        kind ;
        CORBA_any                   value ;
        }
        CORBA_Contained_Description ;

typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_RepositoryId          id ;
        CORBA_RepositoryId          defined_in ;
        CORBA_VersionSpec           version ;
        CORBA_TypeCode              result ;
        CORBA_OperationMode         mode ;
        CORBA_ContextIdSeq          contexts ;
        CORBA_ParDescriptionSeq     parameters ;
        CORBA_ExcDescriptionSeq     exceptions ;
        }
        CORBA_OperationDescription ;

CORBA_Contained_Description     CORBA_OperationDef_describe
                                        ( CORBA_OperationDef  operationdef1,
                                        CORBA_Environment     * ev1 ) ;
```

Arguments

operationdef1	*(in)* the operation definition object
ev1	*(in/out)* the CORBA Environment

Returns

a description of the operation definition object

Exceptions

(standard)

Description

Returns a description of an operation definition object. The ***value*** within the description is the ***CORBA::OperationDescription*** data structure.

Notes

1. Inherited from ***CORBA::Contained***. Other interfaces which inherit from ***CORBA::Contained*** may return different structures for the ***value*** element of ***CORBA::Contained::Description***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

CORBA, 6.5.21.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::destroy - destroy an operation definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::exceptions - exceptions of an operation

Synopsis - C

```
typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_ExceptionDef              * _buffer ;
        }
        CORBA_ExceptionDefSeq ;

CORBA_ExceptionDefSeq       * CORBA_OperationDef__get_exceptions
                                ( CORBA_OperationDef        operationdef1,
                                CORBA_Environment           * ev1 ) ;

void                        CORBA_OperationDef__set_params
                                ( CORBA_OperationDef        operationdef1,
                                CORBA_ExceptionDefSeq       * exceptions1,
                                CORBA_Environment           * ev1 ) ;
```

Arguments

operationdef1	*(in)* the operation definition object
exceptions1	*(in)* the new exceptions
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_OperationDef__get_exceptions(): the exceptions of the operation
CORBA_OperationDef__set_exceptions(): *(void)*

Exceptions

(standard)

Description

Return or set the exceptions of an operation.

Notes

(none)

Reference

CORBA, 6.5.21.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::id - global identifier of an operation definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::mode - mode of an operation

Synopsis - C

```
CORBA_OperationMode     CORBA_OperationDef__get_mode
                            ( CORBA_OperationDef        operationdef1,
                            CORBA_Environment           * ev1 ) ;
void                    CORBA_OperationDef__set_mode
                            ( CORBA_OperationDef        operationdef1,
                            CORBA_OperationMode         * mode1,
                            CORBA_Environment           * ev1 ) ;
```

Arguments

operationdef1	*(in)* the operation definition object
mode1	*(in)* the new operation mode (either ***CORBA_OP_NORMAL*** or ***CORBA_OP_ONEWAY***)
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_OperationDef__get_mode(): the operation mode (either ***CORBA_OP_NORMAL*** or ***CORBA_OP_ONEWAY***)
CORBA_OperationDef__set_mode(): *(void)*

Exceptions

(standard)

Description

Return or set the mode of an operation.

Notes

(none)

Reference

CORBA, 6.5.21.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::move - move operation definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::OperationDef::name - local identifier of an operation definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::params - parameters of an operation

Synopsis - C

```
typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_TypeCode              type ;
        CORBA_IDLType               type_def ;
        CORBA_ParameterMode         mode ;
        }
        CORBA_ParameterDescription ;

typedef struct
        {
        unsigned long               _maximum ;
        unsigned long               _length ;
        CORBA_ParameterDescription  * _buffer ;
        }
        CORBA_ParDescriptionSeq ;

CORBA_ParDescriptionSeq     * CORBA_OperationDef__get_params
                                ( CORBA_OperationDef        operationdef1,
                                CORBA_Environment           * ev1 ) ;
void                        CORBA_OperationDef__set_params
                                ( CORBA_OperationDef        operationdef1,
                                CORBA_ParDescriptionSeq     * parameters1,
                                CORBA_Environment           * ev1 ) ;
```

Arguments

operationdef1	*(in)* the operation definition object
parameters1	*(in)* the new parameters
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_OperationDef__get_params(): the parameters of the operation
CORBA_OperationDef__set_params(): *(void)*

Exceptions

(standard)

Description

Return or set the parameters of an operation.

Notes

1. ***CORBA_ParameterDescription.mode*** has one of the values ***CORBA_PARAM_IN***, ***CORBA_PARAM_OUT***, or ***CORBA_PARAM_INOUT***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

2. When setting the ***params*** attribute, the value of ***CORBA_ParameterDescription.type*** should always be set to ***TC_void***. The value of the ***type_def*** member will cause that of the ***type*** member to be updated automatically.

Reference

CORBA, 6.5.21.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::result - type code of operation result

Synopsis - C

```
CORBA_TypeCode          CORBA_OperationDef__get_result
                            ( CORBA_OperationDef        operationdef1,
                            CORBA_Environment           * ev1 ) ;
```

Arguments

operationdef1	*(in)* the operation definition object
ev1	*(in/out)* the CORBA Environment

Returns

the type code of the operation result

Exceptions

(standard)

Description

Returns the type code of an operation result.

Notes

1. Although the ***CORBA::OperationDef::result*** attribute is ***readonly***, it can be modified indirectly by setting the ***CORBA::OperationDef::result_def*** attribute (*q.v.*).

Reference

CORBA, 6.5.21.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::result_def - IDL type of operation result

Synopsis - C

```
CORBA_IDLType     CORBA_OperationDef__get_result_def
                                ( CORBA_OperationDef     operationdef1,
                                  CORBA_Environment      * ev1 ) ;
void              CORBA_OperationDef__set_result_def
                                ( CORBA_OperationDef     operationdef1,
                                  CORBA_IDLType          typedef1,
                                  CORBA_Environment      * ev1 ) ;
```

Arguments

operationdef1	*(in)* the operation definition object
typedef1	*(in)* the new IDL type object
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_OperationDef__get_result_def(): the IDL type object of the operation result
CORBA_OperationDef__set_result_def(): *(void)*

Exceptions

(standard)

Description

Return or set the IDL type object of an operation result.

Notes

1. Setting the ***result_def*** attribute causes the ***result*** attribute to be modified.

Reference

CORBA, 6.5.21.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::OperationDef::version - get version of an operation definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_ORB_BOA_init ()- initialize Basic Object Adaptor

Synopsis - C

```
#include "orb.h"

typedef CORBA_string    CORBA_ORB_OAid ;

CORBA_BOA               CORBA_ORB_BOA_init
                            ( CORBA_ORB             orb1,
                              int                   * argc,
                              char                  ** argv,
                              CORBA_ORB_OAid        boa_identifier,
                              CORBA_Environment     * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
argc	*(in/out)* number of arguments in ***argv***
argv	*(in/out)* list of character string arguments
boa_identifier	*(in)* specific BOA to be initialized
ev1	*(in/out)* the CORBA Environment

Returns

the object reference to the BOA

Exceptions

(standard)

Description

Initializes a Basic Object Adaptor (BOA).

Notes

1. Must be invoked separately for each thread or process using the BOA.

2. In this implementation, the ***boa_identifier*** must be *"egypt_001"*.

3. The initial ORB reference ***orb1*** is obtained from ***CORBA_ORB_init ()***.

4. Parameters in ***argv*** must be of the form *"-OA<suffix> <value>"*. Any parameters consumed by this procedure will be removed from ***argv***, and ***argc*** will be adjusted accordingly. (The pointer will be removed from the ***argv*** list, but the corresponding storage will not be deleted.)

5. This implementation defines no meaningful ***argv*** parameters except "-OAid", whose value (if present) overrides that of ***boa_identifier***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

CORBA, 7.5 and 14.26.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_alias_tc - create an alias type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_alias_tc
                        ( CORBA_ORB             orb1,
                        CORBA_RepositoryId      id1,
                        CORBA_Identifier        name1,
                        CORBA_TypeCode          * original_type1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
id1	*(in)* repository identifier of new typecode
name1	*(in)* name of new alias
original_type1	*(in)* original type upon which the alias is based
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Creates a new alias typecode object.

Notes

(none)

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_array_tc - create an array type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_array_tc
                        ( CORBA_ORB                 orb1,
                        CORBA_unsigned_long         length1,
                        CORBA_TypeCode              element_type1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
length1	*(in)* the number of elements in the array
element_type1	*(in)* the type of the array elements
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Creates a new array typecode object.

Notes

(none)

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_enum_tc - create an enumeration type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_enum_tc
                        ( CORBA_ORB                 orb1,
                        CORBA_RepositoryId          id1,
                        CORBA_Identifier            name1,
                        CORBA_EnumMemberSeq         * members1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
id1	*(in)* repository identifier of new typecode
name1	*(in)* name of new enumeration
members1	*(in)* members of new enumeration
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Creates a new enumeration typecode object.

Notes

(none)

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_exception_tc - create an exception type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_exception_tc
                        ( CORBA_ORB                 orb1,
                        CORBA_RepositoryId          id1,
                        CORBA_Identifier            name1,
                        CORBA_StructMemberSeq       * members1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
id1	*(in)* repository identifier of new typecode
name1	*(in)* name of new exception
members1	*(in)* members of new exception
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Creates a new exception typecode object.

Notes

1. The value of ***type_def*** in each element of ***members1*** should be set to ***CORBA_OBJECT_NIL***.

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_interface_tc - create an interface type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_interface_tc
                        ( CORBA_ORB             orb1,
                        CORBA_RepositoryId      id1,
                        CORBA_Identifier        name1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
id1	*(in)* repository identifier of new typecode
name1	*(in)* name of new interface
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Creates a new interface typecode object.

Notes

(none)

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_list - allocate a named-value list

Synopsis - C

```
typedef CORBA_Object          CORBA_NVList ;
typedef CORBA_unsigned_long   CORBA_Status ;

CORBA_Status   CORBA_ORB_create_list ( CORBA_ORB orb1,
                        CORBA_long count1, CORBA_NVList *list1,
                        CORBA_Environment * ev1 ) ;
```

Arguments

orb1	(in) the basic object adaptor belonging to the implementation
count1	(in) the number of entries to be allocated in the list
list1	(out) new named-value list
ev1	(in/out) the CORBA Environment

Returns

the operation status

Exceptions

(standard)

Description

.Allocates a named-value list object and clears it for initial use.

Notes

(none)

Reference

CORBA, 4.4.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_operation_list - create initialized named-value list

Synopsis - C

```
typedef CORBA_Object          CORBA_NVList ;
typedef CORBA_Object          CORBA_OperationDef ;
typedef CORBA_unsigned_long   CORBA_Status ;

void    CORBA_ORB_create_operation_list ( CORBA_ORB orb1,
                CORBA_OperationDef def1,
                CORBA_NVList list1,
                CORBA_Environment * ev1 ) ;
```

Arguments

orb1	(in) the object request broker to create the list
def1	(in) definition of the operation
list1	(out) initialized named-value list
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Returns a named-value list initialized with the argument descriptions for a given operation.

Notes

(none)

Reference

CORBA, 4.4.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_recursive_sequence_tc - create recursive sequence type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_recursive_sequence_tc
                        ( CORBA_ORB                 orb1,
                        CORBA_unsigned_long         bound1,
                        CORBA_unsigned_long         offset1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
bound1	*(in)*
offset1	*(in)*
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Used to create typecodes describing recursive sequences.

Notes

1. xxxxx

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_sequence_tc - create a sequence type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_sequence_tc
                        ( CORBA_ORB             orb1,
                        CORBA_unsigned_long     bound1,
                        CORBA_TypeCode          element_type1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
id1	*(in)* repository identifier of new typecode
bound1	*(in)* the maximum number of elements of the new sequence
element_type1	*(in)* sequence element typecode
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Creates a new sequence typecode object.

Notes

(none)

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_string_tc - create a bounded string type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_string_tc
                        ( CORBA_ORB                 orb1,
                        CORBA_unsigned_long         bound1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
bound1	*(in)* maximum length of string
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Creates a new bounded string typecode object.

Notes

(none)

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_struct_tc - create structure type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_struct_tc
                        ( CORBA_ORB                 orb1,
                        CORBA_RepositoryId          id1,
                        CORBA_Identifier            name1,
                        CORBA_StructMemberSeq       * members1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
id1	*(in)* repository identifier of new typecode
name1	*(in)* name of new structure
members1	*(in)* members of new structure
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Creates a new structure typecode object.

Notes

1. The value of ***type_def*** in each element of ***members1*** should be set to ***CORBA_OBJECT_NIL***.

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::create_union_tc - create union type code

Synopsis - C

```
CORBA_TypeCode      CORBA_ORB_create_union_tc
                        ( CORBA_ORB                 orb1,
                        CORBA_RepositoryId          id1,
                        CORBA_Identifier            name1,
                        CORBA_TypeCode              discriminator_type1,
                        CORBA_UnionMemberSeq        * members1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

orb1	*(in)* reference to the ORB performing the operation
id1	*(in)* repository identifier of new typecode
name1	*(in)* name of new union
discriminator_type1	*(in)* typecode of union discriminator
members1	*(in)* members of new union
ev1	*(in/out)* the CORBA Environment

Returns

the new typecode object

Exceptions

(standard)

Description

Creates a new union typecode object.

Notes

1. The value of ***type_def*** in each element of ***members1*** should be set to ***CORBA_OBJECT_NIL***.

Reference

CORBA, 6.7.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::get_default_context - return reference to default process context

Synopsis - C

```
typedef CORBA_unsigned_long   CORBA_Status ;
typedef CORBA_Object          CORBA_Context ;

CORBA_Status   CORBA_ORB_default_context ( CORBA_ORB orb1,
                        CORBA_Context * ctx1,
                        CORBA_Environment * ev1 ) ;
```

Arguments

orb1	(in) the object request broker
ctx1	(out) the default process context object
ev1	(in/out) the CORBA Environment

Returns

the operation status

Exceptions

(standard)

Description

Obtain the default process context object.

Notes

(none)

Reference

CORBA, 4.6.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_ORB_init ()- initialize ORB

Synopsis - C

```
#include "orb.h"

typedef CORBA_string    CORBA_ORBid ;

CORBA_ORB               CORBA_ORB_init
                            ( int                   * argc,
                            char                    ** argv,
                            CORBA_ORBid             orb_identifier,
                            CORBA_Environment       * ev1 ) ;
```

Arguments

argc	*(in/out)* number of arguments in ***argv***
argv	*(in/out)* list of character string arguments
orb_identifier	*(in)* specific ORB to be initialized
ev1	*(in/out)* the CORBA Environment

Returns

the object reference to the ORB

Exceptions

(standard)

Description

Initializes an Object Request Broker (ORB).

Notes

1. Must be invoked separately for each thread or process using the ORB.
2. In this implementation, the ***orb_identifier*** must be *"egypt_001"*.
3. The initial CORBA environment is obtained from ***eg_environment_create()***.
4. Parameters in ***argv*** must be of the form *"-ORB<suffix> <value>"*. Any parameters consumed by this procedure will be removed from ***argv***, and ***argc*** will be adjusted accordingly. (The pointer will be removed from the ***argv*** list, but the corresponding storage will not be deleted.)
5. This implementation defines no meaningful ***argv*** parameters except "-ORBid", whose value (if present) overrides that of ***orb_identifier***.

Reference

CORBA, 7.4 and 14.26.1.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_ORB_list_initial_services ()- return initial services

Synopsis - C

```
#include "orb.h"

typedef CORBA_string                    CORBA_ORB_ObjectId ;

typedef struct
        {
        unsigned long                   _maximum ;
        unsigned long                   _length ;
        CORBA_ORB_ObjectId              *_buffer ;
        }
        CORBA_sequence_CORBA_ORB_ObjectId ;
typedef CORBA_sequence_CORBA_ORB_ObjectId       CORBA_ORB_ObjectIdList ;

CORBA_ORB_ObjectIdList      CORBA_ORB_list_initial_services
                                ( CORBA_ORB                 orb1,
                                CORBA_Environment           *ev1 ) ;
```

Arguments

orb1	*(in)* an object request broker
ev1	*(in/out)* the CORBA Environment

Returns

a list of the initial services

Exceptions

(standard)

Description

Obtains the initial services available from the ORB.

Notes

1. Each object id in the returned sequence can be passed to ***CORBA_ORB_resolve_initial_references()*** to obtain its object reference for subsequent use.

2. The initial reference to the ORB (***orb1***) is obtained from ***CORBA_ORB_init()***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

3. The values currently defined for initial services are:
   InterfaceRepository
   NameService
   POACurrent
   RootPOA
   SecurityCurrent
   TradingService
   TransactionCurrent

   Further retrieval must be done using those two services.

Reference

CORBA, 7.6 and 14.27. Modified by 97-05-15, 2.6.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::object_to_string - convert an object reference to a string

Synopsis - C

```
CORBA_string    CORBA_ORB_object_to_string ( CORBA_ORB orb1,
                                             CORBA_Object obj1,
                                             CORBA_Environment * ev1 ) ;
```

Arguments

orb1	(in) an object request broker
obj1	(in) the object reference to be converted
ev1	(in/out) the CORBA Environment

Returns

a string form of the object reference

Exceptions

(standard)

Description

Converts an object reference to a string.

Notes

1. The string can be converted back into an object reference using ***CORBA::ORB::string_to_object***. However, the string is guaranteed to be meaningful only to the same ORB which provided it in the first place. The string has no standard format and should be considered opaque.

2. A string version of an object reference can be stored , transmitted, or otherwise used as necessary by applications.

Reference

CORBA, 7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::perform_work - perform unit of work

Synopsis - C

```
void        CORBA_ORB_perform_work ( CORBA_ORB orb1,
                                     CORBA_Environment * ev1 ) ;
```

Arguments

orb1	*(in)* the object request broker
ev1	*(in/out)* the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

If called by the main thread, this operation performs an implementation-defined unit of work. Otherwise, it does nothing.

Notes

1. Used to support threaded environments.

Reference

CORBA, 7.7, as extended by 97-05-15, 2.6.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_ORB_resolve_initial_references ()- resolve initial object references

Synopsis - C

```
#include "orb.h"

typedef CORBA_string            CORBA_ORB_ObjectId ;

CORBA_Object    CORBA_ORB_resolve_initial_references
                        ( CORBA_ORB             orb1,
                        CORBA_ORB_ObjectId      identifier1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

orb1	*(in)* an object request broker
identifier1	*(in)* the name of an initial service
ev1	*(in/out)* the CORBA Environment

Returns

the object reference for the given initial service

Exceptions

CORBA::ORB::InvalidName - unknown value for ***identifier1***
(also standard exceptions)

Description

Returns an object reference for a specified service.

Notes

1. The initial reference to the ORB (***orb1***) is obtained from ***CORBA_ORB_init()***.

2. Values for ***identifier1*** (the names of the initial services) are obtained from the sequence of service names returned by ***CORBA_ORB_list_initial_services ()***.

Reference

CORBA, 7.6 and 14.27.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::run - return when ORB has shut down

Synopsis - C

```
void        CORBA_ORB_run ( CORBA_ORB orb1,
                            CORBA_Environment * ev1 ) ;
```

Arguments

orb1	*(in)* the object request broker
ev1	*(in/out)* the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Returns when the ORB has shut down.

Notes

1. Used to support threaded environments.
2. Gives the main thread to the ORB. Can be used when there are no other activities which require the main thread.
3. Ensures the main thread will not terminate until the ORB has finished its work.

Reference

CORBA, 7.7, as extended by 97-05-15, 2.6.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::shutdown - instruct ORB to shut down

Synopsis - C

```
void        CORBA_ORB_shutdown ( CORBA_ORB orb1,
                                 CORBA_boolean wait_for_completion1,
                                 CORBA_Environment * ev1 ) ;
```

Arguments

orb1	*(in)* the object request broker
wait_for_completion1	*(in)* if ***TRUE***, causes the call to block until the operation is finished
ev1	*(in/out)* the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Instructs the ORB to shut down, also causing all object adaptors to shut down.

Notes

1. Used to support threaded environments.
2. If ***wait_for_completion1*** is ***TRUE***, then the call will block until all ORB operations are completed. This includes adaptor operations and request processing.

Reference

CORBA, 7.7, as extended by 97-05-15, 2.6.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::string_to_object - convert string to an object reference

Synopsis - C

```
CORBA_Object   CORBA_ORB_string_to_object ( CORBA_ORB orb1,
                                            CORBA_string str1,
                                            CORBA_Environment * ev1 ) ;
```

Arguments

orb1	(in) the object request broker to perform the conversion
str1	(in) the string form of the object reference
ev1	(in/out) the CORBA Environment

Returns

an object reference corresponding to the string

Exceptions

(standard)

Description

Converts a string to an object reference. The string must have been returned by ***CORBA::ORB:object_to_string***.

Notes

1. This call must be made to the same ORB used when calling ***CORBA::ORB::object_to_string***. In general, such strings are opaque and not portable among different ORBs.

Reference

CORBA, 7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::ORB::work_pending - determin if ORB needs main thread

Synopsis - C

```
CORBA_boolean       CORBA_ORB_work_pending ( CORBA_ORB orb1,
                                             CORBA_Environment * ev1 ) ;
```

Arguments

orb1	(in) the object request broker
ev1	(in/out) the CORBA Environment

Returns

TRUE, if the ORB needs the main thread to perform some work
FALSE, if the ORB does not need the main thread to perform some work

Exceptions

(standard)

Description

Determines if the ORB needs the main thread to perform some work.

Notes

1. Used to support threaded environments.

Reference

CORBA, 7.7, as extended by 97-05-15, 2.6.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::NVList::add_item - add item to named value list

Synopsis - C

```
CORBA_Status    CORBA_NVList_add_item ( CORBA_NVList nvlist1,
                                        CORBA_Identifier * item_name1,
                                        CORBA_TypeCode * item_type1,
                                        void * value1,
                                        unsigned long value_length1,
                                        CORBA_Flags item_flags1,
                                        CORBA_Environment * ev1 ) ;
```

Arguments

nvlist1	*(in)* the named value list
item_name1	*(in)* name of the new item
item_type1	*(in)* type of the new item
value1	*(in)* value of the new item
value_length1	*(in)* size of ***value1*** (see note 1)
flags1	*(in)* item flags (see note 2)
ev1	*(in/out)* the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Add a new item to a named-value list.

Notes

1. For objects, ***value_length1*** is one. Otherwise, ***value_length1*** is the actual storage size of ***value1*** (using the C ***sizeof*** operator).

2. The flags1 argument can contain the following flags:
   CORBA_ARG_IN - the argument is input-only
   CORBA_ARG_OUT - the argument is output only
   CORBA_ARG_INOUT - the argument is both input and output
   CORBA_IN_COPY_VALUE - a copy of the argument is made and used instead of the original
   CORBA_DEPENDENT_LIST - indicates the associated value (which must be a sublist) should be freed when the parent list is freed

Reference

CORBA, 4.4.1.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::NVList::free - destroy named value list

Synopsis - C

```
CORBA_Status    CORBA_NVList_free ( CORBA_NVList nvlist1,
                                    CORBA_Environment * ev1 ) ;
```

Arguments

nvlist1	*(in)* the named value list
ev1	*(in/out)* the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Destroys a named value list. All associated storage is freed, including any dynamically-allocated out-arg memory.

Notes

1. This function makes an implicit call to ***CORBA::NVList::free_memory***.

Reference

CORBA, 4.4.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::NVList::free_memory - free dynamically-allocated out-arg memory

Synopsis - C

```
CORBA_Status        CORBA_NVList_free_memory ( CORBA_NVList nvlist1,
                                               CORBA_Environment * ev1 ) ;
```

Arguments

nvlist1	*(in)* the named value list
ev1	*(in/out)* the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Frees any dynamically-allocated out-arg memory associated with a named-value list. The list itself is not freed.

Notes

(none)

Reference

CORBA, 4.4.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::NVList::get_count - return number of items in named value list

Synopsis - C

```
CORBA_Status    CORBA_NVList_get_count ( CORBA_NVList nvlist1,
                                         CORBA_long * count1,
                                         CORBA_Environment * ev1 ) ;
```

Arguments

nvlist1	*(in)* the named value list
count1	*(out)* the number of items allocated for ***nvlist1***
ev1	*(in/out)* the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Determine the total number of items allocated for the specified named-value list.

Notes

(none)

Reference

CORBA, 4.4.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Policy::copy - copy policy object

Synopsis - C

```
CORBA_Policy        CORBA_Policy_copy ( CORBA_Policy policy1,
                                        CORBA_Environment * ev1 ) ;
```

Arguments

policy1	*(in)* the policy object to be copied
ev1	*(in/out)* the CORBA Environment

Returns

a new policy object

Exceptions

(standard)

Description

Copy the specified policy object.

Notes

1. The copy does not retain any relationships that the original had with any domain or object.

Reference

CORBA 2.1, 5.8.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Policy::destroy - destroy policy object

Synopsis - C

```
void        CORBA_Policy_destroy ( CORBA_Policy policy1,
                                   CORBA_Environment * ev1 ) ;
```

Arguments

policy1	*(in)* the policy object to be destroyed
ev1	*(in/out)* the CORBA Environment

Returns

(void)

Exceptions

(standard)

Description

Destroy the specified policy object.

Notes

1. It is the responsibility of the policy object to determine if it can be destroyed.

Reference

CORBA 2.1, 5.8.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Policy::policy_type - type of policy object

Synopsis - C

```
typedef CORBA_unsigned_long   PolicyType ;

CORBA_PolicyType      CORBA_Policy__get_policy_type ( CORBA_Policy policy1,
                                                      CORBA_Environment * env1 ) ;
```

Arguments

policy1	*(in)* the policy object whose type is to be determined
ev1	*(in/out)* the CORBA Environment

Returns

the type of policy object

Exceptions

(standard)

Description

Determines the type of the policy object.

Notes

1. The values of ***PolicyType*** are assigned by OMG.

Reference

CORBA 2.1, 5.8.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::PrimitiveDef::def_kind - get definition kind of a primitive definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::PrimitiveDef::destroy - destroy a primitive definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::PrimitiveDef::kind - kind of a primitive definition

Synopsis - C

```
CORBA_PrimitiveKind   CORBA_PrimitiveDef__get_kind
                          ( CORBA_PrimitiveDef          primitive1,
                          CORBA_Environment             * ev1 ) ;
```

Arguments

primitive1	*(in)* the primitive definition object
ev1	*(in/out)* the CORBA Environment

Returns

the ***kind*** of the primitive definition

Exceptions

(standard)

Description

Returns the ***kind*** attribute of a primitive definition object.

Notes

(none)

Reference

CORBA, 6.5.15.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::PrimitiveDef::type - type of a primitive definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

CORBA::Repository::contents - get list of contained or inherited objects

Inherited from

CORBA::Container

Refer to

CORBA::Container::contents

CORBA::Repository::create_alias - create an alias definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_alias

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::create_array - create an array definition

Synopsis - C

```
CORBA_ArrayDef      CORBA_Repository_create_array
                        ( CORBA_Repository      repository1,
                        CORBA_unsigned_long     length1,
                        CORBA_IDLType           element_type1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

repository1	*(in)* the repository to contain the new array definition
length1	*(in)* the number of elements in the new array
element_type1	*(in)* the element type of the new array
ev1	*(in/out)* the CORBA Environment

Returns

a new array definition

Exceptions

(standard)

Description

Create a new array definition in a repository.

Notes

1. An array definition is an anonymous type. Each array definition may be used to define exactly one other type.

Reference

CORBA, 6.5.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::create_constant - create a constant definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_constant

CORBA::Repository::create_enum - create an enumeration definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_enum

CORBA::Repository::create_interface - create an interface definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_interface

CORBA::Repository::create_module - create a module definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_module

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::create_sequence - create a sequence definition

Synopsis - C

```
CORBA_SequenceDef    CORBA_Repository_create_sequence
                         ( CORBA_Repository          * repository1,
                         CORBA_unsigned_long        bound1,
                         CORBA_IDLType              element_type1,
                         CORBA_Environment           * ev1 ) ;
```

Arguments

repository1	*(in)* the repository to contain the new sequence definition
bound1	*(in)* the maximum number of elements in the new sequence (a value of *zero* indicates the sequence will be unbounded)
element_type1	*(in)* the element type of the new sequence
ev1	*(in/out)* the CORBA Environment

Returns

a new sequence definition

Exceptions

(standard)

Description

Create a new sequence definition in a repository.

Notes

1. A sequence definition is an anonymous type. Each sequence definition may be used to define exactly one other type.

Reference

CORBA, 6.5.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::create_string - create a bounded string definition

Synopsis - C

```
CORBA_StringDef     CORBA_Repository_create_string
                        ( CORBA_Repository          * repository1,
                        CORBA_unsigned_long         bound1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

repository1	*(in)* the repository to contain the new string definition
bound1	*(in)* the maximum number of elements in the new string (must be > *zero*)
ev1	*(in/out)* the CORBA Environment

Returns

a new string definition

Exceptions

(standard)

Description

Create a new bounded string definition in a repository.

Notes

1. A string definition is an anonymous type. Each string definition may be used to define exactly one other type.

2. Unbounded string definitions are primitive definitions obtained using ***CORBA::Repository::get_primitive***.

Reference

CORBA, 6.5.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::create_struct - create a structure definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_struct

CORBA::Repository::create_union - create a union definition

Inherited from

CORBA::Container

Refer to

CORBA::Container::create_union

CORBA::Repository::def_kind - get definition kind of a repository

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::Repository::describe_contents - return description of contents

Inherited from

CORBA::Container

Refer to

CORBA::Container::describe_contents

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::destroy - destroy a repository and its contents

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::get_primitive - return primitive definition

Synopsis - C

```
CORBA_PrimitiveDef    CORBA_Repository_get_primitive
                          ( CORBA_Repository          * repository1,
                          CORBA_PrimitiveKind         kind1,
                          CORBA_Environment           * ev1 ) ;
```

Arguments

repository1	*(in)* the repository to be searched
kind1	*(in)* the kind of the desired primitive
ev1	*(in/out)* the CORBA Environment

Returns

an primitive definition with the specified kind

Exceptions

(standard)

Description

Return a primitive definition from a repository, given the definition's kind.

Notes

1. All primitive definitions are owned by the repository, and are immutable.

Reference

CORBA, 6.5.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::lookup - locate a definition relative to a container

Inherited from

CORBA::Container

Refer to

CORBA::Container::lookup

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::lookup_id - locate an object by its repository id

Synopsis - C

```
CORBA_Contained     CORBA_Repository_lookup_id
                        ( CORBA_Repository          * repository1,
                        CORBA_RepositoryId          search_id1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

repository1	*(in)* the repository to be searched
search_id1	*(in)* the repository identifier of the object to be located
ev1	*(in/out)* the CORBA Environment

Returns

an object reference corresponding to the repository identifier

Exceptions

(standard)

Description

Find an object in a repository, given the object's globally-unique repository identifier.

Notes

1. If the repository does not contain an object with the given identifier, a nil object reference is returned.

Reference

CORBA, 6.5.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Repository::lookup_name - locate an object by name

Inherited from

CORBA::Container

Refer to

CORBA::Container::lookup_name

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Request::add_arg - add an argument to a request

Synopsis - C

```
CORBA_Status   CORBA_Request_add_arg ( CORBA_Request request1,
                                       CORBA_Identifier name1, CORBA_TypeCode type1,
                                       void * value1, CORBA_long length1, CORBA_Flags flags1,
                                       CORBA_Environment * ev1 ) ;
```

Arguments

request1	(in) the request to which the argument is to be added
name1	(in) argument name
type1	(in) argument type
value1	(in) argument value
length1	(in) length or count of argument value
flags1	(in) argument flags
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Add an argument to a request.

flags1 should be one of ***CORBA::ARG_IN***, ***CORBA::ARG_OUT***, or ***CORBA::ARG_INOUT***.

If ***CORBA::ARG_IN*** is set, then the flag ***CORBA::IN_COPY_VALUE*** may also be used. This will cause the operation to make a copy of the request value (***value1***) instead of assuming ownership of the storage.

Notes

(none)

Reference

CORBA, 4.2.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Request::delete - delete a request

Synopsis - C

```
CORBA_Status   CORBA_Request_delete ( CORBA_Request request1,
                                      CORBA_Environment * ev1 ) ;
```

Arguments

request	(in) the request to be deleted
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Delete a request, freeing associated memory.

Notes

(none)

Reference

CORBA, 4.2.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Request::get_response - determine whether a request has completed

Synopsis - C

```
CORBA_Status   CORBA_Request_get_response ( CORBA_Request request1,
                                            CORBA_Flags flags1,
                                            CORBA_Environment * ev1 ) ;
```

Arguments

request1	(in) the request whose status is to be checked
flags1	(in) operation flags
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Determines if a request has completed.

The only value currently defined for ***flags1*** is ***CORBA::RESP_NO_WAIT***. If this flag is set, then the operation returns immediately, whether or not ***request1*** has completed. Otherwise, the operation does not return until ***request1*** has completed.

Notes

(none)

Reference

CORBA, 4.3.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Request::invoke -call the ORB to request a method

Synopsis - C

```
CORBA_Status   CORBA_Request_invoke ( CORBA_Request request1,
                                      CORBA_Flags flags1,
                                      CORBA_Environment * ev1 ) ;
```

Arguments

request1	(in) the request to be made
flags1	(in) operation flags (none currently defined)
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Make a synchronous operation request. The operation does not return until the request is completed.

Notes

(none)

Reference

CORBA, 4.2.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::Request::send - initiate an operation based on a request

Synopsis - C

```
CORBA_Status   CORBA_Request_send ( CORBA_Request request1,
                                    CORBA_Flags flags1,
                                    CORBA_Environment * ev1 ) ;
```

Arguments

request1	(in) the request to be performed
flags1	(in) operation flags
ev1	(in/out) the CORBA Environment

Returns

operation status

Exceptions

(standard)

Description

Make an asynchronous operation request. The operation returns immediately, and does not wait for the request to complete.

Only one value for ***flags1*** is currently defined. If ***CORBA::INV_NO_RESPONSE*** is set, or if the operation is defined to be oneway, then there is no need to call ***CORBA::Request::get_response***. (Note this might cause some errors to go undetected.)

Notes

(none)

Reference

CORBA, 4.3.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::SequenceDef::bound - maximum number of elements in a sequence

Synopsis - C

```
CORBA_unsigned_long     CORBA_SequenceDef__get_bound
                            ( CORBA_SequenceDef         sequencedef1,
                            CORBA_Environment           * ev1 ) ;

void                    CORBA_SequenceDef__set_bound
                            ( CORBA_SequenceDef         sequencedef1,
                            CORBA_unsigned_long         bound1
                            CORBA_Environment           * ev1 ) ;
```

Arguments

sequencedef1	*(in)* the sequence definition object
bound1	*(in)* maximum number of elements in the sequence
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_SequenceDef__get_bound(): the maximum number of elements in the sequence
CORBA_SequenceDef__set_bound(): *(void)*

Exceptions

(standard)

Description

Returns or sets the maximum number of elements in a sequence.

Notes

1. A maximum number of elements ***bound1*** of *zero* indicates an unbounded sequence.

Reference

CORBA, 6.5.17.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::SequenceDef::def_kind - get definition kind of sequence definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::SequenceDef::destroy - destroy a sequence definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::SequenceDef::element_type - type code of sequence element

Synopsis - C

```
CORBA_TypeCode          CORBA_SequenceDef__get_element_type
                            ( CORBA_SequenceDef     sequencedef1,
                            CORBA_Environment       * ev1 ) ;
```

Arguments

sequencedef1	*(in)* the sequence definition object
ev1	*(in/out)* the CORBA Environment

Returns

the typecode of the sequence element

Exceptions

(standard)

Description

Return the type code of the sequence element.

Notes

1. This attribute is ***readonly*** and cannot be set. However, it can be changed indirectly by setting the ***element_type_def*** attribute.

Reference

CORBA, 6.5.17.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::SequenceDef::element_type_def - IDL type of sequence element

Synopsis - C

```
CORBA_IDLType       CORBA_SequenceDef__get_element_type_def
                        ( CORBA_SequenceDef     sequencedef1,
                        CORBA_Environment       * ev1 ) ;

void                CORBA_SequenceDef__set_element_type_def
                        ( CORBA_SequenceDef     sequencedef1,
                        CORBA_TypeDef           element1
                        CORBA_Environment       * ev1 ) ;
```

Arguments

sequencedef1	*(in)* the sequence definition object
element1	*(in)* the sequence element IDL type object
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_SequenceDef__get_element_type_def(): the sequence element IDL type
CORBA_SequenceDef__set_element_type_def(): *(void)*

Exceptions

(standard)

Description

Returns or sets the IDL type of a sequence element.

Notes

1. Setting this attribute will automatically set the ***element_type*** attribute.

Reference

CORBA, 6.5.17.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::SequenceDef::type - type of a sequence definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_ServerRequest_ctx ()- determine passed context values

14.24.1

CORBA_ServerRequest_exceptions ()- report exceptions to client

14.24.1

CORBA_ServerRequest_op_name ()- return name of operation being performed

14.24.1

CORBA_ServerRequest_params ()- retrieve parameters from ServerRequest

14.24.1

CORBA_ServerRequest_result ()- report result value for an operation

14.24.1

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA_string_alloc () - allocate memory for string

Synopsis - C

```
char    * CORBA_string_alloc ( unsigned long length ) ;
```

Arguments

length	(in) number of characters in string (exclusive of trailing *NUL*)

Returns

pointer to the storage area if successful, else ***NULL***

Exceptions

(standard)

Description

Allocates storage for data of type ***CORBA::string***.

Notes

1. The storage must be freed using ***CORBA_free ()***.

Reference

CORBA, 14.12.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::StringDef::bound - maximum length of a bounded string

Synopsis - C

```
CORBA_unsigned_long     CORBA_StringDef__get_bound
                            ( CORBA_StringDef       stringdef1,
                            CORBA_Environment       * ev1 ) ;

void                    CORBA_StringDef__set_bound
                            ( CORBA_StringDef       stringdef1,
                            CORBA_unsigned_long     bound1
                            CORBA_Environment       * ev1 ) ;
```

Arguments

stringdef1	*(in)* the string definition object
bound1	*(in)* maximum length of the bounded string
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_StringDef__get_bound(): the maximum length of the bounded string
CORBA_StringDef__set_bound(): *(void)*

Exceptions

(standard)

Description

Returns or sets the maximum length of a bounded string.

Notes

1. The maximum length ***bound1*** must not be set to *zero*. A value of *zero* indicates an unbounded string, which is represented as a ***CORBA::PrimitiveDef***.

Reference

CORBA, 6.5.16.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::StringDef::def_kind - get definition kind of a string definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::StringDef::destroy - destroy a string definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::StringDef::type - type of a string definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

CORBA::StructDef::absolute_name - get scoped name of structure definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::absolute_name

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::StructDef::containing_repository - get repository of structure definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::containing_repository

CORBA::StructDef::def_kind - get definition kind of a structure definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::StructDef::defined_in - get container of structure definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

CORBA::StructDef::describe - describe structure definition

Inherited from

CORBA::TypedefDef

Refer to

CORBA::TypedefDef::describe

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::StructDef::destroy - destroy a structure definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::StructDef::id - global identifier of a structure definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::StructDef::members - members of a structure definition

Synopsis - C

```
typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_TypeCode              type ;
        CORBA_IDLType               type_def ;
        }
        CORBA_StructMember ;

typedef
        {
        unsigned long               _maximum ;
        unsigned long               _length ;
        CORBA_StructMember          * _buffer ;
        }
        CORBA_StructMemberSeq ;

CORBA_StructMemberSeq       * CORBA_StructDef__get_members
                                ( CORBA_StructDef           structdef1,
                                CORBA_Environment           * ev1 ) ;

void                        CORBA_StructDef__set_members
                                ( CORBA_StructDef           structdef1,
                                CORBA_StructMemberSeq       * members1,
                                CORBA_Environment           * ev1 ) ;
```

Arguments

structdef1	*(in)* the structure definition object
members1	*(in)* new members descriptions
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_StructDef__get_members(): member description sequence
CORBA_StructDef__set_members(): *(void)*

Exceptions

(standard exceptions)

Description

Returns or sets the members of a structure definition.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Notes

1. When setting the ***members*** attribute (using ***CORBA_StructDef__set_members()***), the ***type*** member of each ***CORBA::StructMember*** structure should be set to ***TC_void***. The value of the ***type_def*** member will automatically cause that of the ***type*** member value to be updated correctly.

Reference

CORBA, 6.5.10.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::StructDef::move - move structure definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::StructDef::name - local identifier of a structure definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

CORBA::StructDef::type - type of a structure definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

CORBA::StructDef::version - get version of a structure definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::content_type - return type code of type code content

Synopsis - C

```
#include "orb.h"

CORBA_TypeCode      CORBA_TypeCode_content_type
                        ( CORBA_TypeCode        typecode1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose content type is to be known
ev1	*(in/out)* the CORBA Environment

Returns

the ***CORBA::TypeCode*** object which describes the content

Exceptions

CORBA::TypeCode::BadKind
(also standard exceptions)

Description

Returns the ***TypeCode*** object describing the content of a sequence, array, or alias ***TypeCode*** object.

Notes

1. If ***typecode1*** is not a sequence, array, or alias typecode object, then the ***BadKind*** exception will be raised.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::default_index - return default index of a union typecode

Synopsis - C

```
#include "orb.h"

CORBA_long    CORBA_TypeCode_default_index
                              ( CORBA_TypeCode          typecode1,
                              CORBA_Environment         * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose default index is to be known
ev1	*(in/out)* the CORBA Environment

Returns

the default index of the type code

Exceptions

CORBA::TypeCode::BadKind
(also standard exceptions)

Description

Returns the default index of a union ***TypeCode*** object. The default index is the index of the default member. If there is no default member, then *-1* is returned.

Notes

1. If ***typecode1*** is not a union typecode object, then the ***BadKind*** exception will be raised.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::discriminator_type - return type code of a union discriminator

Synopsis - C

```
#include "orb.h"

CORBA_TypeCode      CORBA_TypeCode_discriminator_type
                        ( CORBA_TypeCode        typecode1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose discriminator type is to be known
ev1	*(in/out)* the CORBA Environment

Returns

the type code of the discriminator

Exceptions

CORBA::TypeCode::BadKind
(also standard exceptions)

Description

Returns the type code of the discriminator of a union ***TypeCode*** object.

Notes

1. If ***typecode1*** is not a union typecode object, then the ***BadKind*** exception will be raised.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::equal - compare two type codes

Synopsis - C

```
#include "orb.h"

CORBA_boolean       CORBA_TypeCode_equal
                        ( CORBA_TypeCode        typecode1,
                        CORBA_TypeCode          typecode2,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the first typecode to be compared
typecode2	*(in)* the second typecode to be compared
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_TRUE if the two typecodes are equal, else ***CORBA_FALSE***

Exceptions

(standard exceptions)

Description

Compares two typecodes to see if they are equal. Two typecodes are considered equal if they give identical results when any of the ***CORBA::TypeCode*** operations are applied to them.

Notes

(none)

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::id - return repository id of a type code

Synopsis - C

```
#include "orb.h"

CORBA_string    CORBA_TypeCode_id
                                ( CORBA_TypeCode        typecode1,
                                CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose repository id is to be known
ev1	*(in/out)* the CORBA Environment

Returns

the repository id of the type code

Exceptions

CORBA::TypeCode::BadKind
(also standard exceptions)

Description

Returns the repository id of an object reference, structure, union, enumeration, alias, or exception ***TypeCode*** object.

Notes

1. If ***typecode1*** is not a object reference, structure, union, enumeration, alias, or exception typecode object, then the ***BadKind*** exception will be raised.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::kind - return kind of type code

Synopsis - C

```
#include "orb.h"

CORBA_TCKind        CORBA_TypeCode_content_type
                        ( CORBA_TypeCode        typecode1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose kind is to be known
ev1	*(in/out)* the CORBA Environment

Returns

the kind of the type code

Exceptions

(standard exceptions)

Description

Returns the kind of a ***TypeCode*** object.

Notes

(none)

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::length - return length or bound from a type code

Synopsis - C

```
#include "orb.h"

CORBA_unsigned_long  CORBA_TypeCode_length
                         ( CORBA_TypeCode        typecode1,
                         CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose length or bound is to be known
ev1	*(in/out)* the CORBA Environment

Returns

the number of elements or bound of the type code

Exceptions

CORBA::TypeCode::BadKind
(also standard exceptions)

Description

Returns the number of elements of an array ***TypeCode*** object. Returns the bound of a sequence or string ***TypeCode*** object.

Notes

1. If ***typecode1*** is not an array, sequence, or string typecode object, then the ***BadKind*** exception will be raised.

2. If a sequence or string is unbounded, then *zero* will be returned.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::member_count - return number of type code members

Synopsis - C

```
#include "orb.h"

CORBA_unsigned_long  CORBA_TypeCode_member_count
                         ( CORBA_TypeCode          typecode1,
                         CORBA_Environment         * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose member count is to be known
ev1	*(in/out)* the CORBA Environment

Returns

the number of members of the type code

Exceptions

CORBA::TypeCode::BadKind
(also standard exceptions)

Description

Returns the number of members in a structure, union, enumeration or exception ***TypeCode*** object.

Notes

1. If ***typecode1*** is not a structure, union, enumeration, or exception typecode object, then the ***BadKind*** exception will be raised.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::member_label - return label of a union type code member

Synopsis - C

```
#include "orb.h"

CORBA_any       CORBA_TypeCode_member_label
                    ( CORBA_TypeCode        typecode1,
                    unsigned long           index1,
                    CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose member count is to be known
index1	*(in)* the index (beginning with zero) of the member
ev1	*(in/out)* the CORBA Environment

Returns

the value of the type code member's label

Exceptions

CORBA::TypeCode::BadKind - typecode does not represent a union
CORBA::TypeCode::Bounds - the value of index1 exceeds that of the highest-numbered member
(also standard exceptions)

Description

Returns the value of the label of the specified member in a union ***TypeCode*** object.

Notes

1. If ***typecode1*** is not a union typecode object, then the ***BadKind*** exception will be raised.

2. If ***index1*** ≥ the number of members in the typecode, then the ***Bounds*** exception will be raised.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::member_name - return name of a type code member

Synopsis - C

```
#include "orb.h"

CORBA_string        CORBA_TypeCode_member_name
                        ( CORBA_TypeCode        typecode1,
                        unsigned long           index1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose member count is to be known
index1	*(in)* the index (beginning with zero) of the member
ev1	*(in/out)* the CORBA Environment

Returns

the name of the specified member of the type code

Exceptions

CORBA::TypeCode::BadKind - typecode does not represent a structure, union, enumeration, or exception

CORBA::TypeCode::Bounds - the value of index1 exceeds that of the highest-numbered member

(also standard exceptions)

Description

Returns the name of the specified member in a structure, union, enumeration or exception ***TypeCode*** object.

Notes

1. If ***typecode1*** is not a structure, union, enumeration, or exception typecode object, then the ***BadKind*** exception will be raised.

2. If ***index1*** ≥ the number of members in the typecode, then the ***Bounds*** exception will be raised.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::member_type - return type code of a type code member

Synopsis - C

```
#include "orb.h"

CORBA_TypeCode      CORBA_TypeCode_member_type
                        ( CORBA_TypeCode        typecode1,
                        unsigned long           index1,
                        CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose type is to be known
index1	*(in)* the index (beginning with zero) of the member
ev1	*(in/out)* the CORBA Environment

Returns

the type code of the specified member of the type code

Exceptions

CORBA::TypeCode::BadKind - typecode does not represent a structure, union, or exception

CORBA::TypeCode::Bounds - the value of index1 exceeds that of the highest-numbered member

(also standard exceptions)

Description

Returns the typecode of the specified member of a structure, union or exception ***TypeCode*** object.

Notes

1. If ***typecode1*** is not a structure, union, or exception typecode object, then the ***BadKind*** exception will be raised.

2. If ***index1*** ≥ the number of members in the typecode, then the ***Bounds*** exception will be raised.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::name - return local name of a type code

Synopsis - C

```
#include "orb.h"

CORBA_string    CORBA_TypeCode_name
                                ( CORBA_TypeCode        typecode1,
                                CORBA_Environment       * ev1 ) ;
```

Arguments

typecode1	*(in)* the typecode object whose name is to be known
ev1	*(in/out)* the CORBA Environment

Returns

the local name of the type code

Exceptions

CORBA::TypeCode::BadKind
(also standard exceptions)

Description

Returns the simple name which identifies an object reference, structure, union, enumeration, alias, or exception ***TypeCode*** object within its enclosing scope.

Notes

1. If ***typecode1*** is not a object reference, structure, union, enumeration, alias, or exception typecode object, then the ***BadKind*** exception will be raised.

Reference

CORBA, 6.7.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypeCode::param_count - return parameter count of a type code

Note

This operation is deprecated by the CORBA specification, and is no longer supported. Its function has been assumed by other operations and attributes.

CORBA::TypeCode::parameters - return parameters of a type code

Note

This operation is deprecated by the CORBA specification, and is no longer supported. Its function has been assumed by other operations and attributes.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypedefDef::absolute_name - get scoped name of typedef definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::absolute_name

CORBA::TypedefDef::containing_repository - get repository of typedef definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::containing_repository

CORBA::TypedefDef::def_kind - get definition kind of a typedef definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::TypedefDef::defined_in - get container of typedef definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypedefDef::describe - describe typedef definition

Synopsis - C

```
typedef struct
        {
        CORBA_DefinitionKind        kind ;
        CORBA_any                   value ;
        }
        CORBA_Contained_Description ;

typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_RepositoryId          id ;
        CORBA_RepositoryId          defined_in ;
        CORBA_VersionSpec           version ;
        CORBA_TypeCode              type ;
        }
        CORBA_TypeDescription ;

CORBA_Contained_Description     CORBA_TypedefDef_describe
                                        ( CORBA_TypedefDef    typedefdef1,
                                        CORBA_Environment     * ev1 ) ;
```

Arguments

typedefdef1	*(in)* the typedef definition object
ev1	*(in/out)* the CORBA Environment

Returns

a description of the typedef definition object

Exceptions

(standard)

Description

Returns a description of a typedef definition object. The ***value*** within the description is the ***CORBA::TypeDescription*** data structure.

Notes

1. The following interfaces inherit from ***CORBA::TypedefDef***, and also return a ***CORBA::TypeDescription*** in the ***value*** element of ***CORBA::Contained::Description***:
   CORBA::AliasDef
   CORBA::EnumDef
   CORBA::StructDef
   CORBA::UnionDef.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

2. Inherited from ***CORBA::Contained***. Other interfaces which inherit from ***CORBA::Contained*** may return different structures for the ***value*** element of ***CORBA::Contained::Description***.

Reference

CORBA, 6.5.9.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypedefDef::destroy - destroy a typedef definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

CORBA::TypedefDef::id - global identifier of a typedef definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

CORBA::TypedefDef::move - move typedef definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::TypedefDef::name - local identifier of a typedef definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::TypedefDef::type - type of a typedef definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

CORBA::TypedefDef::version - get version of a typedef definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

CORBA::UnionDef::absolute_name - get scoped name of union definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::absolute_name

CORBA::UnionDef::containing_repository - get repository of union definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::containing_repository

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::UnionDef::def_kind - get definition kind of a union definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::def_kind

CORBA::UnionDef::defined_in - get container of union definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::defined_in

CORBA::UnionDef::describe - describe union definition

Inherited from

CORBA::TypedefDef

Refer to

CORBA::TypedefDef::describe

CORBA::UnionDef::destroy - destroy a union definition

Inherited from

CORBA::IRObject

Refer to

CORBA::IRObject::destroy

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::UnionDef::discriminator_type - type code of union discriminator

Synopsis - C

```
CORBA_TypeCode      * CORBA_UnionDef__get_discriminator_type
                                ( CORBA_UnionDef          uniondef1,
                                  CORBA_Environment       * ev1 ) ;
```

Arguments

uniondef1	*(in)* the union definition object
ev1	*(in/out)* the CORBA Environment

Returns

typecode of the union's discriminator

Exceptions

(standard exceptions)

Description

Returns the type code of the union's discriminator.

Notes

1. Although ***CORBA::UnionDef::discriminator_type*** is a ***readonly*** attribute, its value can be modified indirectly by setting ***CORBA::UnionDef::discriminator_type_def***.

Reference

CORBA, 6.5.11.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::UnionDef::discriminator_type_def - IDL type of union discriminator

Synopsis - C

```
CORBA_IDLType       * CORBA_UnionDef__get_discriminator_type_def
                                ( CORBA_UnionDef          uniondef1,
                                  CORBA_Environment       * ev1 ) ;

void                CORBA_UnionDef__set_discriminator_type_def
                                ( CORBA_UnionDef          uniondef1,
                                  CORBA_IDLType           type_def1,
                                  CORBA_Environment       * ev1 ) ;
```

Arguments

uniondef1	*(in)* the union definition object
type_def1	*(in)* new IDL type of union's discriminator
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_UnionDef__get_discriminator_type_def(): IDL type of union's discriminator
CORBA_UnionDef__set_discriminator_type_def(): *(void)*

Exceptions

(standard exceptions)

Description

Returns or sets the IDL type of a union's discriminator.

Notes

1. Setting this value will automatically cause the value of ***discriminator_type*** to be updated.

Reference

CORBA, 6.5.11.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::UnionDef::id - global identifier of a union definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::id

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::UnionDef::members - members of a union definition

Synopsis - C

```
typedef struct
        {
        CORBA_Identifier            name ;
        CORBA_any                   label ;
        CORBA_TypeCode              type ;
        CORBA_IDLType               type_def ;
        }
        CORBA_UnionMember ;

typedef
        {
        unsigned long               _maximum ;
        unsigned long               _length ;
        CORBA_UnionMember           * _buffer ;
        }
        CORBA_UnionMemberSeq ;

CORBA_UnionMemberSeq    * CORBA_UnionDef__get_members
                        ( CORBA_UnionDef            uniondef1,
                        CORBA_Environment           * ev1 ) ;

void                    CORBA_UnionDef__set_members
                        ( CORBA_UnionDef            uniondef1,
                        CORBA_UnionMemberSeq        * members1,
                        CORBA_Environment           * ev1 ) ;
```

Arguments

uniondef1	*(in)* the union definition object
members1	*(in)* new members descriptions
ev1	*(in/out)* the CORBA Environment

Returns

CORBA_UnionDef__get_members(): member description sequence
CORBA_UnionDef__set_members(): *(void)*

Exceptions

(standard exceptions)

Description

Returns or sets the members of a union definition.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Notes

1. When setting the ***members*** attribute (using ***CORBA_UnionDef__set_members()***), the ***type*** member of each ***CORBA::UnionMember*** structure should be set to ***TC_void***. The value of the ***type_def*** member will automatically cause that of the ***type*** member value to be updated correctly.

Reference

CORBA, 6.5.11.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

CORBA::UnionDef::move - move union definition to new container

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::move

CORBA::UnionDef::name - local identifier of a union definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::name

CORBA::UnionDef::type - type of a union definition

Inherited from

CORBA::IDLType

Refer to

CORBA::IDLType::type

CORBA::UnionDef::version - get version of a union definition

Inherited from

CORBA::Contained

Refer to

CORBA::Contained::version

©1997 Bionic Buffalo Corporation; All Rights Reserved.

eg_environment_create - create CORBA environment

Synopsis - C

```
#include "orb.h"

typedef struct
        {
        unsigned long           length ;        /* number of bytes in value */
        unsigned char           * value ;
        }
        eg_key_t ;

typedef struct
        {
        unsigned long           flags ;
        eg_key_t                machine_key ;
        eg_key_t                session_key ;
        eg_key_t                thread_key ;
        char                    * thread_name ;
        }
        eg_environment_parms_t ;

unsigned long           eg_environment_create
                                ( eg_environment_parms_t        * parms1,
                                CORBA_Environment               ** ev1 ) ;
```

Arguments

parms1	*(in)* various initialization parameters
ev1	*(out)* the CORBA Environment

Returns

zero if success, else a non-zero failure code

Exceptions

(standard)

Description

Creates the CORBA environment structure for the C language mapping.

Notes

1. Must be invoked separately for each thread or process using the ORB.

2. The user-defined parameter ***machine_key*** must be unique on the network for every machine. One possibility is to use the internet address of the machine.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

3. The user-defined parameter ***session_key*** must be unique on a given machine for every time the machine is rebooted. One possibility is to use the time of day and date of the last reboot. Another option is a simple numbering of the sessions.

4. The user-defined parameter ***thread_key*** must be unique on a given machine and session for every thread or process. If thread or process identifiers are reused, then one possibility is to use the time of day and date of thread creation with a system-designated thread or process identifier. If threads or process identifiers are never reused until the system is rebooted, then the thread or process identifier alone might be used.

5. The primary purpose of ***machine_key***, ***session_key***, and ***thread_key*** is the generation of unique object identifiers. CORBA requires that all object identifiers be globally unique throughout all time.

6. The ***thread_name*** must be unique for the thread at any given time. A specific value of ***thread_name*** may be reused as long as the previous thread or process with the same name is terminated.

7. No values are currently defined for ***flags***.

8. Ownership of memory for the environment parameters structure is retained by the caller.

Reference

proprietary definition

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

eg_register_panic_handler - register user-provided routine for panic handling

Synopsis - C

```
#include "orb.h"

typedef void     ( eg_panic_handler_t )
                        ( int                   fault_code,
                        long                    fault_parms1 [ 8 ],
                        void                    * fault_parms2 [ 4 ],
                        char                    * thread_name,
                        void                    * fault_location,
                        char                    * procedure_name ) ;

unsigned long   eg_register_panic_handler
                        ( eg_panic_handler_t    * proc1 ) ;
```

Arguments

proc1	*(in)* user-provided panic handler routine

Returns

zero if success, else a non-zero failure code

Exceptions

(standard)

Description

Register a panic handler to be invoked in case of unrecoverable error.

Notes

1. A panic routine is invoked only when the normal recovery or exception mechanisms are inadequate or unavailable, or when their use might create additional error conditions. For example, if there is insufficient memory to allocate an exception data structure, then a panic routine will be called.

2. The panic handling routine designated by ***proc1*** should terminate the process or thread and not return to the caller.

3. Although the ***fault_code*** passed to the handler will always be valid, in some circumstances the other parameters may not always be meaningful. If a parameter is not meaningful, it will be set to ***NULL*** or zero as appropriate.

4. If a panic routine is not provided by the user, then the default behaviour will be to terminate the process or thread, returning (if possible in the environment) the ***fault_code*** as an exit or return value.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

5. These are the defined values for ***fault_code***:

Fault Code	Meaning
EG_FAULT_CODE_MEMORY	insufficient memory to continue
EG_FAULT_CODE_SYSTEM	unrecoverable error from operating system or other component of the operating environment
EG_FAULT_CODE_USER	the user has signalled an unrecoverable fault
EG_FAULT_CODE_LIMITS	configured operational limits exceeded
EG_FAULT_CODE_SECURITY	unrecoverable threat to system security
EG_FAULT_CODE_OTHER	other unrecoverable error

Reference

proprietary definition

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::AdapterActivator::unknown_adaptor - create required POA

Synopsis - C

```
CORBA_boolean    PortableServer_AdapterActivator_unknown_adaptor
                     ( PortableServer_AdapterActivator    activator1,
                     PortableServer_POA                   parent_poa1,
                     CORBA_string                         name1,
                     CORBA_Environment                    * env1 ) ;
```

Arguments

activator1	*(in)* the activator associated with the POA to be created
parent_poa1	*(in)* the parent of the POA to be created
name1	*(in)* the name of the POA to be created
env1	*(in/out)* the CORBA environment

Returns

TRUE, if the required POA was created
FALSE, if the required POA was not created

Exceptions

(standard exceptions)

Description

Called by the ORB when the POA associated with a requested object does not exist, causing the required POA to be created.

Notes

1. If several POAs must be created, then this routine will be called successively for each parent POA until all necessary child POAs have been created.

2. If this operation returns ***FALSE***, then the ORB will fail the request by raising the ***CORBA::OBJECT_NOT_EXIST*** exception.

3. May be called by the ORB in the process of satisfying an incoming request. May also be called if the ***PortableServer::POA::find_POA*** operation is invoked with ***activate_it = TRUE***.

Reference

97-05-15, 3.3.3.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::activate_object - generate object reference and activate object

Synopsis - C

```
PortableServer_ObjectId
                    PortableServer_POA_activate_object
                        ( PortableServer_POA          poa1,
                        PortableServer_Servant        servant1,
                        CORBA_Environment             * env1 ) ;
```

Arguments

poa1	*(in)* the POA
servant1	*(in)* the object's servant
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::WrongPolicy - either ***SYSTEM_ID*** or ***RETAIN*** policy is not in effect
PortableServer::ServantAlreadyActive - the ***UNIQUE_ID*** policy is in effect and the servant is already in the active object map
(also standard exceptions)

Description

Generates an object identifier for the specified servant, and places an entry into the active object map.

Notes

(none)

Reference

97-05-15, 3.3.8.14.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::activate_object_with_id - activate object

Synopsis - C

```
void        PortableServer_POA_activate_object_with_id
                    ( PortableServer_POA        poa1,
                    PortableServer_ObjectId     * object_id1,
                    PortableServer_Servant      servant1,
                    CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
object_id1	*(in)* the object's identifier
servant1	*(in)* the object's servant
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::WrongPolicy - either ***SYSTEM_ID*** or ***RETAIN*** policy is not in effect
PortableServer::ServantAlreadyActive - the ***UNIQUE_ID*** policy is in effect and the servant is already in the active object map
PortableServer::ObjectAlreadyActive - an object with the given value of ***object_id1*** is already in the active object map
(also standard exceptions)

Description

Associates a servant with an object identifier, and places an entry into the active object map.

Notes

(none)

Reference

97-05-15, 3.3.8.15.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_POA - create newchild POA

Synopsis - C

```
PortableServer_POA    PortableServer_POA_create_POA
                          ( PortableServer_POA      poa1,
                          CORBA_string              adaptor_name1,
                          CORBA_POAManager          poa_manager1,
                          CORBA_PolicyList          * policies1,
                          CORBA_Environment         * env1 ) ;
```

Arguments

poa1	*(in)* the parent POA
adaptor_name1	*(in)* name of new child POA
poa_manager1	*(in)* the POA manager to be associated with the new POA
policies1	*(in)* a list of policies to be applied to the new POA
env1	*(in/out)* the CORBA environment

Returns

the new POA

Exceptions

PortableServer::AdaptorAlreadyExists - the parent POA already has a child with the same name

PortableServer::InvalidPolicy - either:

(1) one or more of the policy objects given are not valid for the ORB implementation; or,

(2) two or more of the policy objects are mutually incompatible; or

(3) one or more of the policy objects requires administrative action which has not yet been performed

(also standard exceptions)

Description

Create a new POA as achild of the target POA.

Notes

1. Policy objects are not inherited by a child POA from the parent POA.

2. The members of ***policies1*** are copied by this call, so an application is free to destroy or reuse the list after this call has returned, or while the new POA is active.

3. If ***poa_manager1*** is ***NULL***, then a new POA manager is created for the new POA.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

97-05-15, 3.3.8.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_id_assignment_policy - create an id assignment policy object

Synopsis - C

```
PortableServer_IdAssignmentPolicy
        PortableServer_POA_create_id_assignment_policy
                ( PortableServer_POA                    poa1,
                PortableServer_IdAssignmentPolicyValue  value1,
                CORBA_Environment                       * env1 ) ;
```

Arguments

poa1	*(in)* the POA
value1	*(in)* the value of the new policy
env1	*(in/out)* the CORBA environment

Returns

the new policy object

Exceptions

(standard exceptions)

Description

Create a new id assignment policy object with the specified value.

Notes

1. The application must call ***PortableServer::IdAssignmentPolicy::destroy*** when the policy object is no longer needed.

2. The new policy object is passed to ***PortableServer::POA::create_POA*** for application to a new POA.

3. Allowable id assignment policy values are:

    - ***PortableServer_USER_ID***: Object IDs are assigned by the application.

    - ***PortableServer_SYSTEM_ID***: Object IDs are assigned by the POA.

4. If no id uniqueness policy object is supplied to a POA, then the default policy is ***PortableServer_SYSTEM_ID***.

5. System-assigned IDs of persistent objects must be unique across all instantiations of the same POA.

Reference

97-05-15, 3.3.8.5.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_id_uniqueness_policy - create an id uniqueness policy object

Synopsis - C

```
PortableServer_IdUniquenessPolicy
        PortableServer_POA_create_id_uniqueness_policy
                ( PortableServer_POA                    poa1,
                PortableServer_IdUniquenessPolicyValue  value1,
                CORBA_Environment                       * env1 ) ;
```

Arguments

poa1	*(in)* the POA
value1	*(in)* the value of the new policy
env1	*(in/out)* the CORBA environment

Returns

the new policy object

Exceptions

(standard exceptions)

Description

Create a new id uniqueness policy object with the specified value.

Notes

1. The application must call ***PortableServer::IdUniquenssPolicy::destroy*** when the policy object is no longer needed.

2. The new policy object is passed to ***PortableServer::POA::create_POA*** for application to a new POA.

3. Allowable id uniqueness policy values are:

    - ***PortableServer_UNIQUE_ID***: Each servant supports a single object ID.

    - ***PortableServer_MULTIPLE_ID***: Each servant may support more than one object ID.

4. If no id uniqueness policy object is supplied to a POA, then the default policy is ***PortableServer_UNIQUE_ID***.

Reference

97-05-15, 3.3.8.5.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_implicit_activation_policy - create an implicit activation policy object

Synopsis - C

```
PortableServer_ImplicitActivationPolicy
        PortableServer_POA_create_implicit_activation_policy
                ( PortableServer_POA                        poa1,
                PortableServer_ImplicitActivationPolicyValue value1,
                CORBA_Environment                           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
value1	*(in)* the value of the new policy
env1	*(in/out)* the CORBA environment

Returns

the new policy object

Exceptions

(standard exceptions)

Description

Create a new implicit activation policy object with the specified value.

Notes

1. The application must call ***PortableServer::ImplicitActivationPolicy::destroy*** when the policy object is no longer needed.

2. The new policy object is passed to ***PortableServer::POA::create_POA*** for application to a new POA.

3. Allowable implicit activation policy values are:

   - ***PortableServer_IMPLICIT_ACTIVATION***: The POA supports implicit activation of servants. (The servant retention policy must be ***PortableServer_RETAIN***, and the ID assignment policy must be ***PortableServer_SYSTEM_ID***.)

   - ***PortableServer_NO_IMPLICIT_ACTIVATION***: implicit activation of servants is not supported.

4. If no implicit activation policy object is supplied to a POA, then the default policy is ***PortableServer_NO_IMPLICIT_ACTIVATION***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

97-05-15, 3.3.8.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_lifespan_policy - create a lifespan policy object

Synopsis - C

```
PortableServer_LifespanPolicy
        PortableServer_POA_create_lifespan_policy
                ( PortableServer_POA                poa1,
                PortableServer_LifespanPolicyValue  value1,
                CORBA_Environment                   * env1 ) ;
```

Arguments

poa1	*(in)* the POA
value1	*(in)* the value of the new policy
env1	*(in/out)* the CORBA environment

Returns

the new policy object

Exceptions

(standard exceptions)

Description

Create a new lifespan policy object with the specified value.

Notes

1. The application must call ***PortableServer::LifespanPolicy::destroy*** when the policy object is no longer needed.

2. The new policy object is passed to ***PortableServer::POA::create_POA*** for application to a new POA.

3. Allowable lifespan policy values are:

    - ***PortableServer_TRANSIENT***: Objects cannot outlive the process in which they are first created.

    - ***PortableServer_PERSISTENT***: Objects can outlive the process in which they are created.

4. If no lifespan policy object is supplied to a POA, then the default policy is ***PortableServer_TRANSIENT***.

5. Persistent objects are associated with the POA which created them. If that POA is not active when the object is needed, then the ORB will attempt to activate it.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

97-05-15, 3.3.8.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_reference - create object reference

Synopsis - C

```
CORBA_Object    PortableServer_POA_create_reference
                    ( PortableServer_POA        poa1,
                    CORBA_RepositoryId          interface1,
                    CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
interface1	*(in)* interface repository ID for new object reference
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::WrongPolicy - the ***SYSTEM_ID*** policy is not in effect
(also standard exceptions)

Description

Generates an object identifier. There is no activation, and no entry is placed into the active object map.

Notes

1. The object's identifier can be retrieved using ***PortableServer::POA::reference_to_id***.

Reference

97-05-15, 3.3.8.17.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_reference_with_id - create object reference

Synopsis - C

```
CORBA_Object    PortableServer_POA_create_reference
                    ( PortableServer_POA        poa1,
                    PortableServer_ObjectId     object_id1,
                    CORBA_RepositoryId          interface1,
                    CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
object_id1	*(in)* new object identifier
interface1	*(in)* interface repository ID for new object reference
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard exceptions)

Description

Generates an object identifier. There is no activation, and no entry is placed into the active object map.

Notes

1. If the ***PortableServer_SYSTEM_ID*** policy is in effect, then the use of an object identifier not created by the system for the same POA may raise the ***CORBA::BAD_PARAM*** exception.

Reference

97-05-15, 3.3.8.19.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_request_processing_policy - create a request processing policy object

Synopsis - C

```
PortableServer_RequestProcessingPolicy
        PortableServer_POA_create_request_processing_policy
                ( PortableServer_POA                        poa1,
                PortableServer_RequestProcessingPolicyValue value1,
                CORBA_Environment                           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
value1	*(in)* the value of the new policy
env1	*(in/out)* the CORBA environment

Returns

the new policy object

Exceptions

(standard exceptions)

Description

Create a new request processing policy object with the specified value.

Notes

1. The application must call ***PortableServer::RequestProcessingPolicy::destroy*** when the policy object is no longer needed.

2. The new policy object is passed to ***PortableServer::POA::create_POA*** for application to a new POA.

3. Allowable request processing policy values are:

   - ***PortableServer_USE_ACTIVE_OBJECT_MAP_ONLY***: The object ID must be found in the active object map, otherwise a ***CORBA::OBJECT_NOT_EXIST*** exception is generated. (The servant retention policy must be ***PortableServer_RETAIN***.)

   - ***PortableServer_USE_DEFAULT_SERVANT***: If the object is not found in the active object map, then the default servant (if any) is used. (The ID uniqueness policy must be ***PortableServer_MULTIPLE_ID***.)

   - ***PortableServer_USE_SERVANT_MANAGER***: If the object is not found in the active object map, then the servant manager (if any) is used to locate a servant.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

4. If no id uniqueness policy object is supplied to a POA, then the default policy is ***PortableServer_USE_ACTIVE_OBJECT_MAP_ONLY***.

Reference

97-05-15, 3.3.8.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_servant_retention_policy - create a servant retention policy object

Synopsis - C

```
PortableServer_ServantRetentionPolicy
        PortableServer_POA_create_servant_retention_policy
                ( PortableServer_POA                        poa1,
                PortableServer_ServantRetentionPolicyValue  value1,
                CORBA_Environment                           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
value1	*(in)* the value of the new policy
env1	*(in/out)* the CORBA environment

Returns

the new policy object

Exceptions

(standard exceptions)

Description

Create a new servant retention policy object with the specified value.

Notes

1. The application must call ***PortableServer::ServantRetentionPolicy::destroy*** when the policy object is no longer needed.

2. The new policy object is passed to ***PortableServer::POA::create_POA*** for application to a new POA.

3. Allowable servant retention policy values are:

    - ***PortableServer_RETAIN***: Active servants are retained in the active object map.

    - ***PortableServer_NON_RETAIN***: Servants are not retained by the POA in the active object map.

4. If no id uniqueness policy object is supplied to a POA, then the default policy is ***PortableServer_RETAIN***.

5. If the ***NON_RETAIN*** policy is in effect, then the request processing policy must be either ***USE_DEFAULT_SERVANT*** or ***USE_SERVANT_MANAGER***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

97-05-15, 3.3.8.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::create_thread_policy - create a thread policy object

Synopsis - C

```
PortableServer_ThreadPolicy
        PortableServer_POA_create_thread_policy
                ( PortableServer_POA                poa1,
                PortableServer_ThreadPolicyValue    value1,
                CORBA_Environment                   * env1 ) ;
```

Arguments

poa1	*(in)* the POA
value1	*(in)* the value of the new policy
env1	*(in/out)* the CORBA environment

Returns

the new policy object

Exceptions

(standard exceptions)

Description

Create a new thread policy object with the specified value.

Notes

1. The application must call ***PortableServer::ThreadPolicy::destroy*** when the policy object is no longer needed.

2. The new policy object is passed to ***PortableServer::POA::create_POA*** for application to a new POA.

3. Allowable thread policy values are:

    - ***PortableServer_ORB_CTRL_MODEL***: The ORB assigns incoming requests to threads. Concurrent assignments may occur in a multi-threaded environment.

    - ***PortableServer_SINGLE_THREAD_MODEL***: All requests are processed in a way that is safe for single-threaded code.

4. If no thread policy object is supplied to a POA, then the default policy is ***PortableServer_ORB_CTRL_MODEL***.

5. Use of the single thread model may cause the ORB to use only the main thread. In this case, the application may be required to use ***CORBA::ORB::perform_work*** or ***CORBA::ORB::run*** to make sure the ORB itself acquires the main thread.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

97-05-15, 3.3.8.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::deactivate_object - remove entry from active object map

Synopsis - C

```
void        PortableServer_POA_deactivate_object
                    ( PortableServer_POA        poa1,
                    PortableServer_ObjectId     object_id1,
                    CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
object_identifier1	*(in)* the identifier of the object to be removed from the map
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::WrongPolicy - either ***RETAIN*** policy is not in effect
PortableServer::ObjectNotActive - the object with the given identifier is not in the active object map
(also standard exceptions)

Description

Removes an entry from the active object map.

Notes

1. If a servant manager is associated with the POA, then ***PortableServer::ServantLocator::etherealize*** is invoked. The ***deactivate_object*** operation does not wait for the ***etherealize*** operation to complete before returning. If there are multiple objects using the same servant, then ***etherealize*** may be called multiple times. The servant locator must not destroy a servant which is still associated with active objects.

Reference

97-05-15, 3.3.8.16.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::destroy - destroy a POA and its descendents

Synopsis - C

```
void        PortableServer_POA_destroy
                    ( PortableServer_POA      poa1,
                    CORBA_boolean             etherealize_objects1,
                    CORBA_boolean             wait_for_completion1,
                    CORBA_Environment         * env1 ) ;
```

Arguments

poa1	*(in)* the POA to be destroyed
etherealize_objects1	*(in)* if ***TRUE***, call etherealize operation for each active object (*see Note 1*)
wait_for_completion1	*(in)* if ***TRUE***, wait for operation to complete before returning
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard exceptions)

Description

Destroy a POA and its descendents.

Notes

1. If the following conditions are met:
   (a) the ***etherealize_objects1*** parameter is ***TRUE***; and
   (b) the ***RETAIN*** policy is in effect for the POA; and
   (c) a servant manager is registered with the POA
   then the ***PortableServant::ServantActivator::etherealize*** operation will be called for each active object after the POA has been destroyed.

2. After this call, the same POA may be re-created later in the same process. (Note this does not apply to the ***PortableServer::POAManager::deactivate*** operation, *q.v.*)

Reference

97-05-15, 3.3.8.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::find_POA - seek child POA

Synopsis - C

```
PortableServer_POA    PortableServer_POA_find_POA
                          ( PortableServer_POA      poa1,
                          CORBA_string              adaptor_name1,
                          CORBA_boolean             activate_it1,
                          CORBA_Environment         * env1 ) ;
```

Arguments

poa1	*(in)* the parent POA
adaptor_name1	*(in)* name of child POA
activate_it1	*(in)* if ***TRUE*** and if the sought POA does not exist, then the target's ***AdapterActivator*** is invoked to activate the child POA
env1	*(in/out)* the CORBA environment

Returns

the child POA

Exceptions

PortableServer::AdaptorNonExistent - the child POA could not be found or activated *(also standard exceptions)*

Description

Look up the designated child POA of the target. If it cannot be found and if the ***activate_it1*** parameter is ***TRUE***, then attempt to activate the child by invoking the target's ***AdapterActivator***.

Notes

(none)

Reference

97-05-15, 3.3.8.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::get_servant - get default servant of POA

Synopsis - C

```
PortableServer_Servant
        PortableServer_POA_get_servant
                ( PortableServer_POA        poa1,
                CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
env1	*(in/out)* the CORBA environment

Returns

the POA's default servant

Exceptions

PortableServer::WrongPolicy - the ***USE_DEFAULT_SERVANT*** policy is not in effect
PortableServer::NoServant - a servant has not yet been assigned to the POA
(also standard exceptions)

Description

Get the default servant of the target POA.

Notes

(none)

Reference

97-05-15, 3.3.8.12.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::get_servant_manger - get servant manager of POA

Synopsis - C

```
PortableServer_ServantManager
        PortableServer_POA_get_servant_manager
                ( PortableServer_POA        poa1,
                CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
env1	*(in/out)* the CORBA environment

Returns

the POA's servant manager (***NULL*** if none has been assigned)

Exceptions

PortableServer::WrongPolicy - the ***USE_SERVANT_MANAGER*** policy is not in effect
(also standard exceptions)

Description

Get the servant manager of the target POA.

Notes

1. An application is free to assign its own servant manager to the root POA.

2. The presence or characteristics of the initial servant manager for the root POA is implementation-dependent. A newly-created POA has no adapter activator.

Reference

97-05-15, 3.3.8.10.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::id_to_reference - return reference for given object identifier

Synopsis - C

```
CORBA_Object        PortableServer_POA_id_to_reference
                        ( PortableServer_POA            poa1,
                        PortableServer_ObjectId         object_id1,
                        CORBA_Environment               * env1 ) ;
```

Arguments

poa1	*(in)* the POA
object_identifier1	*(in)* the object identifier for which the reference is sought
env1	*(in/out)* the CORBA environment

Returns

the object reference associated with the given object identifier

Exceptions

PortableServer::WrongPolicy - the ***RETAIN*** policy is not in effect
PortableServer::ObjectNotActive - the given object is not in the current object map
(also standard exceptions)

Description

Returns an object reference for the given object identifier.

Notes

(none)

Reference

97-05-15, 3.3.8.24.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::id_to_servant - return servant for given object identifier

Synopsis - C

```
PortableServer_Servant
        PortableServer_POA_id_to_servant
                ( PortableServer_POA          poa1,
                PortableServer_ObjectId       object_id1,
                CORBA_Environment             * env1 ) ;
```

Arguments

poa1	*(in)* the POA
object_identifier1	*(in)* the object identifier for which the servant is sought
env1	*(in/out)* the CORBA environment

Returns

the servant associated with the identifier

Exceptions

PortableServer::WrongPolicy - the ***RETAIN*** policy is not in effect
PortableServer::ObjectNotActive - the given object is not in the current object map
(also standard exceptions)

Description

Returns the servant for the given object identifier.

Notes

(none)

Reference

97-05-15, 3.3.8.23.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::reference_to_id - return object ID for object reference

Synopsis - C

```
PortableServer_ObjectId
        PortableServer_POA_reference_to_id
                ( PortableServer_POA        poa1,
                CORBA_Object                object1,
                CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
object1	*(in)* the object reference for which the object identifier is sought
env1	*(in/out)* the CORBA environment

Returns

the object identifier associated with the reference

Exceptions

PortableServer::WrongPolicy - *(this exception is permitted by the specification IDL, but is not currently implemented: it exists for future extensions)*
PortableServer::WrongAdaptor - the given object was not created by this POA
(also standard exceptions)

Description

Returns the object identifier for the given object reference.

Notes

(none)

Reference

97-05-15, 3.3.8.22.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::reference_to_servant - return servant for object reference

Synopsis - C

```
PortableServer_Servant
        PortableServer_reference_to_servant
                ( PortableServer_POA        poa1,
                CORBA_Object                object1,
                CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
object1	*(in)* the object reference for which the servant is sought
env1	*(in/out)* the CORBA environment

Returns

the servant associated with the object reference

Exceptions

PortableServer::WrongPolicy - neither the ***RETAIN*** policy nor the ***USE_DEFAULT_SERVANT*** policy is in effect
PortableServer::ObjectNotActive - the required object is not active
PortableServer::WrongAdaptor - the object reference was not created by this POA
(also standard exceptions)

Description

Returns the servant associated with a given object.

If the ***RETAIN*** policy is in effect and the object is in the active object map, then the associated servant is returned.

If the ***USE_DEFAULT_SERVANT*** policy is in effect and a default servant has been registered for the object, then default servant is returned.

If neither of these two conditions applies, then the ***ObjectNotActive*** exception is raised.

Notes

(none)

Reference

97-05-15, 3.3.8.21.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::servant_to_id - return object ID for servant

Synopsis - C

```
PortableServer_ObjectId
        PortableServer_POA_servant_to_id
                ( PortableServer_POA        poa1,
                PortableServer_Servant      servant1,
                CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
servant1	*(in)* the servant associated with the object ID
env1	*(in/out)* the CORBA environment

Returns

the object identifier associated with the servant

Exceptions

PortableServer::WrongPolicy - the ***RETAIN*** policy is not in effect; *OR* one of either the ***UNIQUE_ID*** or ***IMPLICIT_ACTIVATION*** policies is not in effect

PortableServer::ServantNotActive - the required servant is not active or cannot be activated

(also standard exceptions)

Description

Returns an object identifier for the given servant.

If the ***UNIQUE_ID*** policy is in effect and the servant is active, then the associated object identifier is returned.

If the ***IMPLICIT_ACTIVATION*** policy is in effect, then the servant is activated using an object identifier created by the POA, and the associated object identifier is returned.

If neither of these two conditions applies, then the ***ServantNotActive*** exception is raised.

Notes

1. If ***IMPLICIT_ACTIVATION*** is used and the servant is already active, then the ***MULTIPLE_ID*** policy also must be in effect.

Reference

97-05-15, 3.3.8.19.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::servant_to_reference - return object reference for servant

Synopsis - C

```
CORBA_Object    PortableServer_POA_servant_to_reference
                    ( PortableServer_POA        poa1,
                    PortableServer_Servant      servant1,
                    CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
servant1	*(in)* the servant associated with the object reference
env1	*(in/out)* the CORBA environment

Returns

the object reference associated with the servant

Exceptions

PortableServer::WrongPolicy - the ***RETAIN*** policy is not in effect; *OR* one of either the ***UNIQUE_ID*** or ***IMPLICIT_ACTIVATION*** policies is not in effect

PortableServer::ServantNotActive - the required servant is not active or cannot be activated

(also standard exceptions)

Description

Returns an object reference for the given servant.

If the ***UNIQUE_ID*** policy is in effect and the servant is active, then the associated object reference is returned.

If the ***IMPLICIT_ACTIVATION*** policy is in effect, then the servant is activated using an object identifier created by the POA, and the associated object reference is returned.

If neither of these two conditions applies, then the ***ServantNotActive*** exception is raised.

Notes

1. If ***IMPLICIT_ACTIVATION*** is used and the servant is already active, then the ***MULTIPLE_ID*** policy also must be in effect.

Reference

97-05-15, 3.3.8.20.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::set_servant - assign default servant to POA

Synopsis - C

```
void        PortableServer_POA_set_servant
                    ( PortableServer_POA        poa1,
                    PortableServer_Servant      servant1,
                    CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
servantr1	*(in)* the POA's new default servant
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::WrongPolicy - the ***USE_DEFAULT_SERVANT*** policy is not in effect
(also standard exceptions)

Description

Assign a default servant to the target POA. The default servant will be used for all requests on objects which are not in the active servant map.

Notes

(none)

Reference

97-05-15, 3.3.8.13.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::set_servant_manger - assign servant manager to POA

Synopsis - C

```
void        PortableServer_POA_set_servant_manager
                    ( PortableServer_POA                poa1,
                    PortableServer_ServantManager       manager1,
                    CORBA_Environment                   * env1 ) ;
```

Arguments

poa1	*(in)* the POA
manager1	*(in)* the POA's new servant manager
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::WrongPolicy - the ***USE_SERVANT_MANAGER*** policy is not in effect
(also standard exceptions)

Description

Assign a servant manager to the target POA.

Notes

1. An application is free to assign its own servant manager to the root POA.

Reference

97-05-15, 3.3.8.11.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::the_activator - adapter activator of POA

Synopsis - C

```
PortableServer_AdapterActivator
        PortableServer_POA__get_the_activator
                ( PortableServer_POA            poa1,
                CORBA_Environment               * env1 ) ;

void    PortableServer_POA__set_the_activator
                ( PortableServer_POA            poa1,
                PortableServer_AdaptorActivator activator1,
                CORBA_Environment               * env1 ) ;
```

Arguments

poa1	*(in)* the POA
activator1	*(in)* new adapter activator for POA
env1	*(in/out)* the CORBA environment

Returns

PortableServer_POA__get_the_activator: the adapter activator associated with the target
PortableServer_POA__set_the_activator: *(void)*

Exceptions

(standard exceptions)

Description

Return or set the adapter activator associated with the POA.

Notes

1. An application is free to assign its own adapter activator to the root POA.

2. The presence or characteristics of the initial adapter activator for the root POA is implementation-dependent. A newly-created POA has no adapter activator.

Reference

97-05-15, 3.3.8.9.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::the_name - name of POA

Synopsis - C

```
CORBA_string    PortableServer_POA__get_the_name
                                ( PortableServer_POA        poa1,
                                CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
env1	*(in/out)* the CORBA environment

Returns

the name of the POA (in the context of the parent POA)

Exceptions

(standard exceptions)

Description

Return the name of the POA.

Notes

1. Names of children are unique for a given parent.
2. The name of the root POA is implementation-dependent.

Reference

97-05-15, 3.3.8.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::the_parent - parent of POA

Synopsis - C

```
PortableServer_POA    PortableServer_POA__get_the_parent
                          ( PortableServer_POA          poa1,
                          CORBA_Environment             * env1 ) ;
```

Arguments

poa1	*(in)* the POA
env1	*(in/out)* the CORBA environment

Returns

the parent of the POA

Exceptions

(standard exceptions)

Description

Return the parent of the POA.

Notes

1. The parent of the root POA is ***NULL***.

Reference

97-05-15, 3.3.8.7.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POA::the_POAManager - manager of POA

Synopsis - C

```
PortableServer_POAManager    PortableServer_POA__get_the_POAManager
                                    ( PortableServer_POA        poa1,
                                    CORBA_Environment           * env1 ) ;
```

Arguments

poa1	*(in)* the POA
env1	*(in/out)* the CORBA environment

Returns

the POAManager associated with the target

Exceptions

(standard exceptions)

Description

Return the POAManager associated with the POA.

Notes

(none)

Reference

97-05-15, 3.3.8.8.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POAManager::activate - activate POA manager

Synopsis - C

```
void    PortableServer_POAManager_activate
            ( PortableServer_POAManager     poa_manager1,
            CORBA_Environment               * env1 ) ;
```

Arguments

poa_manager1	*(in)* the POA manager to be activated
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::AdapterInactive - the POA manager was in the *inactive* state
(also standard exceptions)

Description

Change the state of a POA manager to *active*, allowing the associated POAs to process requests.

Notes

1. To enter the *active* state, a POA manager must first be in the *discarding* or *holding* state.

2. The maximum number of requests which can be received or enqueued by a POA is defined by the implementation. If this limit is exceeded, then additional requests will be returned with the ***CORBA::TRANSIENT*** exception.

Reference

97-05-15, 3.3.2.3.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POAManager::deactivate - deactivate POA manager

Synopsis - C

```
void    PortableServer_POAManager_deactivate
            ( PortableServer_POAManager     poa_manager1,
            CORBA_boolean                   etherealize_objects1,
            CORBA_boolean                   wait_for_completion1,
            CORBA_Environment               * env1 ) ;
```

Arguments

poa_manager1	*(in)* the POA manager to be deactivated
etherealize_objects1	*(in)* specifies if associated POAs are to etherealize active objects
wait_for_completion1	*(in)* specifies if the operation is to block until all requests are completed
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::AdapterInactive - the POA manager was already in the *inactive* state
(also standard exceptions)

Description

Change the state of a POA manager to *inactive*, causing the associated POAs to reject incoming requests.

Notes

1. If ***etherealize_objects1*** is ***TRUE***, then associated POAs that have the ***RETAIN*** and ***USE_SERVANT_MANAGER*** policies will perform ***etherealize*** on the associated servant manager for all active objects after the POA manager has entered the *inactive* state. (The use of ***FALSE*** for ***etherealize_objects1*** is intended for unrecoverable errors and other crises.)

2. If ***wait_for_completion1*** is ***TRUE***, then the operation will block until:
   (a) there are no active requests in associated POAs, and
   (b) any necessary etherealize operations have completed

3. If ***CORBA::ORB::shutdown*** is called, the ORB invokes ***PortableServer::POAManager::deactivate*** on each known POA manager. The parameters are set as follows:
   (a) ***etherealize_objects1*** is set to ***TRUE***, and
   (b) ***wait_for_completion1*** is set to the value of the equivalent parameter passed to ***CORBA::ORB::shutdown***

©1997 Bionic Buffalo Corporation; All Rights Reserved.

Reference

97-05-15, 3.3.2.6.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POAManager::discard_requests - change POA manager to the *discarding* state

Synopsis - C

```
void    PortableServer_POAManager_discard_requests
            ( PortableServer_POAManager     poa_manager1,
            CORBA_boolean                   wait_for_completion1,
            CORBA_Environment               * env1 ) ;
```

Arguments

poa_manager1	*(in)* the POA manager to be placed in the *discarding* state
wait_for_completion1	*(in)* specifies if the operation is to block until all requests are completed
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::AdapterInactive - the POA manager was in the *inactive* state
(also standard exceptions)

Description

Change the state of a POA manager to *discarding*, causing the associated POAs to discard incoming requests by raising the ***CORBA::TRANSIENT*** exception.

Notes

1. If ***wait_for_completion1*** is ***TRUE***, then the operation will block until:
   (a) there are no active requests in associated POAs, or
   (b) the POA manager is changed to a state other than *discarding*

Reference

97-05-15, 3.3.2.5.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::POAManager::hold_requests - change POA manager to the *holding* state

Synopsis - C

```
void    PortableServer_POAManager_hold_requests
            ( PortableServer_POAManager     poa_manager1,
            CORBA_boolean                   wait_for_completion1,
            CORBA_Environment               * env1 ) ;
```

Arguments

poa_manager1	*(in)* the POA manager to be placed in the *holding* state
wait_for_completion1	*(in)* specifies if the operation is to block until all requests are completed
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

PortableServer::AdapterInactive - the POA manager was in the *inactive* state
(also standard exceptions)

Description

Change the state of a POA manager to *holding*, causing the associated POAs to enqueue incoming requests.

Notes

1. If ***wait_for_completion1*** is ***TRUE***, then the operation will block until:
   (a) there are no active requests in associated POAs, or
   (b) the POA manager is changed to a state other than *holding*

Reference

97-05-15, 3.3.2.4.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::ServantActivator::etherealize - deactivate servant for object

Synopsis - C

```
void    PortableServer_ServantActivator_etherealize
                    ( PortableServer_ServantActivator    activator1,
                    PortableServer_ObjectId              object_id1,
                    PortableServer_Servant               servant1,
                    CORBA_boolean                        cleanup_in_progress1,
                    CORBA_boolean                        remaining_activations1,
                    CORBA_Environment                    * env1 ) ;
```

Arguments

activator1	*(in)* the servant activator
object_id1	*(id)* identifier of object to be deactivated
adaptor1	*(in)* the object's servant
cleanup_in_progress1	*(in)* if ***TRUE***, indicates that the operation was invoked because either ***deactivate*** or ***destroy*** was called with an ***etherealize_objects*** parameter of ***TRUE***; if ***FALSE***, it was invoked for other reasons
remaining_activations1	*(in)* if ***TRUE***, indicates that the servant is associated with other active objects at the time of the call; if ***FALSE***, this is the only active object associated with this servant
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard exceptions)

Description

Deactivates an object.

Notes

1. The POA will invoke this routine when a request is received for an object which is not currently active, and when the ***USE_SERVANT_MANAGER*** and ***RETAIN*** policies are in effect.

2. An active object may be etherealized even if it was not incarnated by the servant manager.

3. This operation is invoked under any of the following circumstances:

    - The object is explicitly deactivated by an invocation of ***PortableServer::POA::deactivate_object***.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

- When the ORB or POA determines that an object must be deactivated based on some policy such as a limitation in the number of simultaneously-active objects or an inactivity timeout.

- ***PortableServer::POAManager::deactivate*** is invoked on a POA manager associated with a POA which has active objects

4. The server's entry in the active object map is removed before this operation is called.

Reference

97-05-15, 3.3.5.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::ServantActivator::incarnate - activate servant for object

Synopsis - C

```
PortableServer_Servant  PortableServer_ServantActivator_incarnate
                                ( PortableServer_ServantActivator   activator1,
                                  PortableServer_ObjectId           object_id1,
                                  PortableServer_POA                adaptor1,
                                  CORBA_Environment                 * env1 ) ;
```

Arguments

activator1	*(in)* the servant activator
object_id1	*(id)* identifier of object to be activated
adaptor1	*(in)* the object's adaptor
env1	*(in/out)* the CORBA environment

Returns

the servant associated with the specified object identifier

Exceptions

PortableServer::ForwardRequest - request should be forwarded to a different object *(also standard exceptions)*

Description

Locate or create an appropriate servant for the specified object identifier.

Notes

1. The POA will invoke this routine when a request is received for an object which is not currently active, and when the ***USE_SERVANT_MANAGER*** and ***RETAIN*** policies are in effect.

2. If the ***UNIQUE_ID*** policy is in effect, then it is an error for the servant activator to return a servant which is already active for a different object. In that case, the POA will return a ***CORBA::OBJ_ADAPTER*** exception. (Note that this policy does not preclude two *different* POAs from using the same servant.)

3. After this call, the servant and object are placed into the active object map. Once the servant is active, then subsequent requests can go directly to the servant (bypassing the servant activator).

Reference

97-05-15, 3.3.5.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::ServantLocator::postinvoke - end one-time use of servant

Synopsis - C

```
void    PortableServer_ServantLocator_postinvoke
                                ( PortableServer_ServantLocator         locator1,
                                  PortableServer_ObjectId               object_id1,
                                  PortableServer_POA                    adaptor1,
                                  CORBA_Identifier                      operation1,
                                  PortableServer_ServantLocator_Cookie  cookie,
                                  CORBA_Environment                     * env1 ) ;
```

Arguments

locator1	*(in)* the servant locator
object_id1	*(id)* identifier of object for operation
adaptor1	*(in)* the object's adaptor
operation1	*(in)* name of operation to be performed
cookie1	*(in)* cookie returned from ***PortableServer::ServantLocator::preinvoke***
env1	*(in/out)* the CORBA environment

Returns

(void)

Exceptions

(standard exceptions)

Description

Ends the one-time use of a servant.

Notes

1. The POA will invoke ***postinvoke*** after calling ***preinvoke*** and after the servant has performed the operation.

Reference

97-05-15, 3.3.6.2.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.

PortableServer::ServantLocator::preinvoke - prepare servant for object

Synopsis - C

```
PortableServer_Servant  PortableServer_ServantLocator_preinvoke
                                ( PortableServer_ServantLocator          locator1,
                                  PortableServer_ObjectId                object_id1,
                                  PortableServer_POA                     adaptor1,
                                  CORBA_Identifier                       operation1,
                                  PortableServer_ServantLocator_Cookie   * cookie,
                                  CORBA_Environment                      * env1 ) ;
```

Arguments

locator1	*(in)* the servant locator
object_id1	*(id)* identifier of object for operation
adaptor1	*(in)* the object's adaptor
operation1	*(in)* name of operation to be performed
cookie1	*(out)* cookie to be passed to ***PortableServer::ServantLocator::postinvoke***
env1	*(in/out)* the CORBA environment

Returns

the servant associated with the specified object identifier

Exceptions

PortableServer::ForwardRequest - request should be forwarded to a different object *(also standard exceptions)*

Description

Prepare a servant for a single use.

Notes

1. The POA will invoke this routine when a request is received for an object which is not active, and when the ***USE_SERVANT_MANAGER*** and ***NON_RETAIN*** policies are in effect.

Reference

97-05-15, 3.3.6.1.

Availability

All CORBA products.

©1997 Bionic Buffalo Corporation; All Rights Reserved.