SOMobjects Developer's Toolkit

Programmer's Reference Volume III: Abstract Interface Definitions

SOMobjects Version 3.0

Note: Before using this information and the product it supports, be sure to read the general information under **Notices** on page iii.

Second Edition (December 1996)

This edition of *Programmer's Reference Volume III: Abstract Interface Definitions* applies to SOMobjects Developer's Toolkit for SOM Version 3.0 and to all subsequent releases of the product until otherwise indicated in new releases or technical newsletters.

Notices

IBM Corporation may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

COPYRIGHT LICENSE: This publication contains printed sample application programs in source language, which illustrate AIX, OS/2, or Windows programming techniques. You may copy and distribute these sample programs in any form without payment to IBM Corporation, for the purposes of developing, using, marketing, or distributing application programs conforming to the AIX, OS/2, or Windows application programming interface.

Each copy of any portion of these sample programs or any derivative work, which is distributed to others, must include a copyright notice as follows: "© (your company name) (current year), All Rights Reserved." However, the following copyright notice protects this documentation under the Copyright Laws of the United States and other countries which prohibit such actions as, but not limited to, copying, distributing, modifying, and making derivative works.

References in this publication to IBM products, program, or services do not imply that IBM Corporation intends to make these available in all countries in which it operates.

Any reference to IBM licensed programs, products, or services is not intended to state or imply that only IBM licensed programs, products, or services can be used. Any functionally-equivalent product, program or service that does not infringe upon any of the IBM Corporation intellectual property rights may be used instead of the IBM Corporation product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM Corporation, are the user's responsibility.

IBM Corporation may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries in writing to the:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594, USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Department 931S
11400 Burnet Road
Austin, Texas 78758 USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Asia-Pacific users can inquire, in writing, to the:

IBM Director of Intellectual Property and Licensing
IBM World Trade Asia Corporation,
2-31 Roppongi 3-chome,
Minato-ku, Tokyo 106, Japan

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks and Acknowledgements

Contents

About This Book

This book explains the abstract interface definitions introduced by the SOMobjects Object Services. The SOMobjects Object Services provide an implementation of standard interfaces defined by the Object Management Group (OMG) and implementations of the interfaces introduced in this book. The SOMobjects Object Services are object-oriented class libraries for managing objects in distributed applications.

Who Should Use This Book

This book is intended for software developers who need to understand the Abstract interfaces introduced by SOMobjects Object Services. Typically this would be someone who intends to provide an implementation of one of the Abstract interfaces.

You will find having the following background helpful:

- Familiarity with the OMG CORBA 1.1 and CORBA IDL specifications
- Familiarity with the OMG Common Object Services, in particular the Naming Service.
- Knowledge of object-oriented principles
- Familiarity with distributed systems management and object management concepts

How This Book Is Organized

This book provides abstract class information on SOMobjects Developer's Toolkit for SOM Version 3.0.

Highlighting

This book uses the following highlighting conventions:

Bold
Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that you select.

Italics
Identifies parameters whose actual names or values you supply. Also identifies new terminology.

`Monospace`
Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Related Publications

The following books contain information about, or related to, SOMobjects Object Services:

- *Common Object Services Specification Volume 1* (OMG Document Number 94-1-1)
- *CORBAservices: Common Object Services Specification* (OMG Document Number 95-3-31)
- *Programmer's Guide for SOM and DSOM*
- *Programmer's Reference for SOM and DSOM*

Explanation of What This Book Describes

The SOMobjects Object Services provide an implementation of standard interfaces defined by the OMG. The “About Programmer’s Reference for Object Services” in *Programmer’s Reference for Object Services* provides a detailed explanation of the relationship between standard interface definitions and SOMobjects Object Services. This includes:

- A description of various approaches to implementing standards
- An explanation of the approach used by the SOMobjects Object Services in providing implementations of the standards.
- An explanation of how the SOMobjects Object Services implementations are documented.

It is highly recommended that you are familiar with the material in that chapter prior to reading this section.

The interfaces introduced in the OMG standards are treated as abstract interface definitions by SOMobjects Object Services. In addition, SOMobjects Object Services introduces some additional abstract interfaces that are extensions and additions to the OMG standard interfaces. These abstract interfaces then have one or more implementations described in the SOMobjects Developer’s Toolkit:

- *Programmer’s Guide for Object Services*
- *Programmer’s Reference for Object Services*

For most users of the SOMobjects Object Services, the description of the implementations provided in the *Programmer’s Guide for Object Services* and *Programmer’s Reference for Object Services* is sufficient to provide the needed information. However, when there is a need to understand the abstract interface definitions associated with the implementations, the documentation of the standards themselves must be referenced. This book covers the documentation for those abstract interface definitions that are not defined by standards, but are introduced by SOMobjects Object Services. More specifically, the following list identifies the documentation that applies for particular circumstances:

- Using or specializing implementations provided by SOMobjects Object Services.
  - *Programmer’s Guide for Object Services*
  - *Programmer’s Reference for Object Services*
- Understanding or providing implementations of abstract interface definitions introduced by the OMG.
  - *Common Object Services Specification Volume 1* (OMG Document Number 94-1-1)
  - *CORBAservices: Common Object Services Specification* (OMG Document Number 95-3-31)
- Understanding or providing implementations of abstract interface definitions introduced by SOMobjects Object Services
  - *Programmer’s Reference for Abstract Interface Definitions*

Chapter 1. Extended Naming Interface Definitions

The **ExtendedNaming** module introduced by SOMobjects Object Services is an extension of the OMG defined **CosNaming** module. Refer to the *Common Object Services Specification Volume 1* (OMG Document Number 94-1-1) for a complete description of the **CosNaming** module. The BNF for the constraint expression is provided in **Appendix A, BNF for Naming Constraint Language** on page 53.

Contents

The ExtendedNaming Module

The **ExtendedNaming** Module defines the **ExtendedNaming::PropertyBindingIterator** interface, **ExtendedNaming::PropertyIterator** interface, and **ExtendedNaming::IndexIterator** interface, including supporting type definitions and exceptions. The **ExtendedNaming::NamingContext** interface provides additional support to the original OMG **CosNaming::NamingContext** interface for the following:

- Binding and setting properties
- Listing and getting properties
- Resolving objects along with their properties
- Removing properties
- Sharing properties
- Searching contexts for objects of certain properties
- Creating indexes for a context
- Administering the capabilities and policies of a context.

Types

The following are defined in the **ExtendedNaming** Module:

- **typedef struct PB {CosNaming::Istring property_name; boolean shareable;} PropertyBinding;**

  This structure, **ExtendedNaming::PropertyBinding**, defines a property name with an indicator of shareability. It does not include the property's value.
- **struct P {PropertyBinding binding; any value;} Property;**

  This structure, **ExtendedNaming::Property**, defines a property name with an indicator of shareability, along with the property's value.
- **struct ID{CosNaming::Istring propert_name; TypeCode property_type; unsigned long distance;} IndexDescriptor;**
- **typedef sequence <CosNaming::Istring> IList;**
- **typedef sequence <PropertyBinding> PropertyBindingList;**
- **typedef sequence <Property> PropertyList;**
- **typedef sequence <IndexDescriptor> IndexDescriptorList;**

Exceptions

There are no user exceptions defined in the **ExtendedNaming** interface.

ExtendedNaming::PropertyBindingIterator Interface

The **ExtendNaming::PropertyBindingIterator** interface provides support for **ExtendedNaming** property binding iteration.

Intended Usage

An instance of this interface is returned through the **ExtendedNaming::ExtendedNamingContext::list_properties** operation if an extended naming context contains more property bindings than the requested number specified on the **ExtendedNaming::ExtendedNamingContext::list_properties** operation. Clients are expected to utilize the provided concrete implementation of **ExtendedNaming** to gain access to this interface. However, subclassed implementations should realize the tight coupling it maintains with the **ExtendedNaming::ExtendedNamingContext::list_properties** operation.

File Stem

xnaming

Directly Inherited Interfaces

SOMObject Class

Indirectly Inherited Interfaces

None.

Types

None.

New Operations

destroy Operation
next_n Operation
next_one Operation

Exceptions

CORBA 1.1 standard exceptions.

destroy Operation

Destroys the iterator.

IDL Syntax

void destroy ( )

Description

Destroys the iterator.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::PropertyBindingIterator Interface

Related Information

list_properties Operation

next_n Operation

Retrieves a specified maximum number of property bindings.

IDL Syntax

boolean next_n (
in unsigned long *howMany*,
out PropertyBindingList *il*);

Description

Returns a specified maximum number of property bindings in the *il* parameter.This operation is used, in standard CORBA fashion, to obtain the next several name-object bindings from the extended naming context with which the targeted **PropertyBindingIterator** is associated. Calling programs should check the return value for decision making for further invocations on the iterator. The operation returns FALSE if there are no more bindings to obtain, indicating to the calling program that it should not invoke the operation again.

Parameters

howMany
maximum number of bindings.

il
The returned **PropertyBindingList**.

Return Value

This operation returns a Boolean value where FALSE indicates to the client that there are no more bindings and where TRUE indicates more bindings exist.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::PropertyBindingIterator Interface

Related Information

list_properties Operation

next_one Operation

Retrieves the next property binding.

IDL Syntax

boolean next_one(out PropertyBinding *pb*);

Description

Returns the next property binding in the *pb* parameter. This operation is used, in standard CORBA fashion, to obtain the next property binding from the extended naming context for which the targeted **PropertyBindingIterator** is associated. Calling programs should check the return value for decision making for further invocations on the iterator. The operation returns FALSE if there are no more bindings to obtain, indicating to the calling program that it should not invoke the operation again.

Parameters

pb
The returned **PropertyBinding**.

Return Value

This operation returns a Boolean value where FALSE indicates to the client that there are no more bindings and where TRUE indicates more bindings exist.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::PropertyBindingIterator Interface

Related Information

list_properties Operation

ExtendedNaming::PropertyIterator Interface

The **ExtendNaming::PropertyIterator** interface provides support for ExtendedNaming property iteration.

Intended Usage

This interface is instantiated and outputted through the **ExtendedNaming::ExtendedNamingContext::get_properties** or **ExtendedNaming::ExtendedNamingContext::get_all_properties** operations if an extended naming context contains more properties than the requested number specified on the **ExtendedNaming::ExtendedNamingContext::get_properties** or **ExtendedNaming::ExtendedNamingContext::get_all_properties** operations. Clients are expected to utilize the provided concrete implementation of ExtendedNaming to gain access to this interface. However, subclassed implementations should realize the tight coupling it maintains with both the **ExtendedNaming::ExtendedNamingContext::get_properties** operation and the **ExtendedNaming::ExtendedNamingContext::get_all_properties** operation.

File Stem

xnaming

Directly Inherited Interfaces

SOMObject Class

Types

None.

New Operations

destroy Operation
next_n Operation
next_one Operation

Exceptions

CORBA 1.1 standard exceptions.

destroy Operation

Destroys the iterator.

IDL Syntax

void destroy ( )

Description

Destroys the iterator.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::PropertyIterator Interface

Related Information

get_properties Operation
get_all_properties Operation

next_n Operation

Retrieves a specified maximum number of properties.

IDL Syntax

boolean next_n (
in unsigned long *howMany*,
out PropertyList *pl*);

Description

Returns a specified maximum number of properties in the *pl* parameter.This operation is used, in standard CORBA fashion, to obtain the next several properties from the extended naming context with which the targeted **PropertyIterator** is associated. Calling programs should check the return value for decision making for further invocations on the iterator. The operation returns FALSE if there are no more bindings to obtain, indicating to the calling program that it should not invoke the operation again.

Parameters

howMany
The maximum number of bindings.

pl
The returned **PropertyList**.

Return Value

This operation returns a Boolean value where FALSE indicates to the client that there are no more bindings and where TRUE indicates more bindings exist.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::PropertyIterator Interface

Related Information

get_properties Operation
get_all_properties Operation

next_one Operation

Retrieves the next property.

IDL Syntax

boolean ::ExtendedNaming::PropertyIterator::next_one(out Property *p*);

Description

Returns the next property in the *p* parameter. This operation is used, in standard CORBA fashion, to obtain the next property from the extended naming context for which the targeted **PropertyIterator** is associated with. Calling programs should check the return value for decision making for further invocations on the iterator. The operation returns FALSE if there are no more bindings to obtain, indicating to the calling program that it should not invoke the operation again.

Parameters

p
The returned **Property**.

Return Value

This operation returns a Boolean value where FALSE indicates to the client that there are no more bindings and where TRUE indicates more bindings exist.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::PropertyIterator Interface

Related Information

get_properties Operation
get_all_properties Operation

ExtendedNaming::IndexIterator Interface

The **ExtendNaming::IndexIterator** interface provides support for ExtendedNaming index iteration.

Intended Usage

This interface interface is instatiated and outputted through the **ExtendedNaming::ExtendedNamingContext::list_indexes** operation if an extended naming context contains more indexes than the requested number specified on the **ExtendedNaming::ExtendedNamingContext::list_indexes** operation. Clients are expected to utilize the provided concrete implementation of **ExtendedNaming** to gain access to this interface. However, subclassed implementations should realize the tight coupling it maintains with the **ExtendedNaming::ExtendedNamingContext::list_indexes** operation.

File Stem

xnaming

Directly Inherited Interfaces

SOMObject Class

Types

None.

New Operations

destroy Operation
next_n Operation
next_one Operation

Exceptions

CORBA 1.1 standard exceptions.

destroy Operation

Destroys the iterator.

IDL Syntax

void destroy ( )

Description

Destroys the iterator.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::IndexIterator Interface

Related Information

list_indexes Operation

next_n Operation

Retrieves a specified maximum number of index descriptors.

IDL Syntax

boolean next_n (
in unsigned long *howMany*,
out IndexDescriptorList *il*);

Description

Returns a specified maximum number of bindings.This operation is used, in standard CORBA fashion, to obtain the next several index descriptors from the extended naming context for which the targeted **IndexIterator** is associated. Calling programs should check the return value for decision making for further invocations on the iterator. The operation returns FALSE if there are no more bindings to obtain, indicating to the calling program that it should not invoke the operation again.

Parameters

howMany
The maximum number of bindings.

il
The returned **IndexDescriptorList**.

Return Value

This operation returns a Boolean value where FALSE indicates to the client that there are no more bindings and where TRUE indicates more bindings exist.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::IndexIterator Interface

Related Information

list_indexes Operation

next_one Operation

Retrieves the next index descriptor.

IDL Syntax

boolean next_one (out IndexDescriptor *p*);

Description

Returns the next index descriptor in the *p* parameter. This operation is used, in standard CORBA fashion, to obtain the next index descriptor from the extended naming context with which the targeted **IndexIterator** is associated. Calling programs should check the return value for decision making for further invocations on the iterator. The operation returns FALSE if there are no more bindings to obtain, indicating to the calling program that it should not invoke the operation again.

Parameters

p
The returned **IndexDescriptor**.

Return Value

This operation returns a Boolean value where FALSE indicates to the client that there are no more bindings and where TRUE indicates more bindings exist.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::IndexIterator Interface

Related Information

list_indexes Operation

ExtendedNaming::ExtendedNamingContext Interface

The **ExtendNaming::ExtendedNamingContext** interface provides support for Extended Naming NamingContexts and extension to the **CosNaming::NamingContext** interface.

Intended Usage

The **ExtendedNamingContext** interface is provided as an abstract interface subclassed from **CosNaming::NamingContext**. This interface provides additional functionality beyond the **CosNaming::NamingContext** interface. See **The ExtendedNaming Module** on page 2 for additional information. Clients are expected to utilize the provided concrete implementation of ExtendedNaming to gain access to this interface. However, clients can also subclass this interface and provide an additional implementation.

File Stem

xnaming

Directly Inherited Interfaces

CosNaming::NamingContext

Indirectly Inherited Interfaces

SOMObject Class

Types

typedef string Constraint; is a string Indicating the search grammar for property searching.

typedef char Strings

New Operations

add_index Operation
add_properties Operation
add_property Operation
bind_context_with_properties Operation
bind_with_properties Operation
find_all Operation
find_any Operation
find_any_name_binding Operation
get_all_properties Operation
get_features_supported Operation
get_properties Operation
get_property Operation
list_indexes Operation
list_properties Operation
rebind_context_with_properties Operation
rebind_with_properties Operation
remove_all_properties Operation
remove_index Operation
remove_properties Operation
remove_property Operation
resolve_with_all_properties Operation
resolve_with_properties Operation
resolve_with_property Operation
_get_allowed_object_types Operation

_get_allowed_property_names Operation
_get_allowed_property_types Operation

Exceptions

- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** indicates that the property name is invalid. A property name with length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::NotSupported indicates that** the implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::ConflictingPropertyName** indicates the property name is in conflict.
- **ExtendedNaming::ExtendedNamingContext::PropertyNotFound{CosNaming Istring property_name;}** indicates that a property was not found.
- **ExtendedNaming::ExtendedNamingContext::NonSharableProperties** indicates that properties were attempted to be shared and are not shareable properties.
- **ExtendedNaming::ExtendedNamingContext::PropertiesNotShared indicates that** properties were not shared.
- **ExtendedNaming::ExtendedNamingContext::IllegalConstraintExpression** indicates that a constraint expression could not be parsed.
- **ExtendedNaming::ExtendedNamingContext::BindingNotFound;** indicates that a requested binding was not found.

add_index Operation

Identifies a property to be indexed.

IDL Syntax

void add_index (in IndexDescriptor *i*);

Description

Identifies a property to be indexed. The index applies to any name-object bindings in the targeted extended naming context or sub-extended naming contexts up to a depth of *distance*, whose property name and property type are specified in *ExtendedNaming::IndexDescriptor i*. If *distance* is set to 0 this operation searches only the targeted context. Any properties added later to bindings in the target extended naming context or relevant sub-extended naming contexts of this property name and type are automatically added to the index.

Parameters

i
The index descriptor to be added.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

ExtendedNaming::ExtendedNamingContext::NotSupported{}; is raised to indicate that implementation does not support this operation.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

add_properties Operation

Adds properties to name-object binding.

IDL Syntax

void add_properties (
in Name *n*,
in PropertyList *s*);

Description

Adds properties to name-object binding. Adds or updates multiple properties, specified in *PropertyList props*, associated with a name-object binding specified by *Name n*, in a target extended naming context. If a property already exists, the property is updated. If a property does not already exist, a new property is associated with the binding (added).

Note: The sharable flag inside a property's **PropertyBinding** has a characteristic of point-in-time. The sharable flag represents whether the property can be shared *at the point in time it is attempted to be shared*. Updating a property with a sharable flag that is different from what was in existence before the update changes not only the restrictions on the updated property, but it can result, for example, in the updated property marked as unshareable, but presently being shared.)

Parameters

n
The **Name** of the name-object binding.

props
The **PropertyList** to be added.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName;** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::ConflictingPropertyName;** is raised to indicate that the property name is in conflict.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

add_property Operation

Adds a property to name-object binding.

IDL Syntax

void add_property (
in Name *n*,
in Property *prop*);

Description

Adds a property to name-object binding. Adds or updates a single property, specified as *prop,* associated with a name-object binding specified by *Name n*, in a target extended naming context. If the property already exists the property is updated with the specified property, *prop*. If the property does not already exist, then specified property is associated with the binding (added).

Note: The sharable flag inside a property's **PropertyBinding** has a characteristic of 'point in time'. The sharable flag represents whether or not the property can be shared at the point in time it is attempted to be shared. Updating a property with a sharable flag that is different from what was in existence before the update changes not only change the restrictions on the updated property, but may result, for example, in the updated property marked unshareable, but presently being shared.)

Parameters

n
The **Name** of the name-object binding.

prop
The **Property** to be added.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::ConflictingPropertyName** is raised to indicate that the property name is in conflict.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

bind_context_with_properties Operation

Creates a name-**NamingContext** object binding and associate properties.

IDL Syntax

void bind_context_with_properties (
in Name ***n*****,**
in ExtendedNamingContext ***obj*****,**
in PropertyList ***props*** **);**

Description

Binds a naming context with properties. Operates just like the **CosNaming::NamingContext::bind_context** operation in that it binds the specified naming context into the target extended naming context. In addition, it defines properties associated with the binding in *PropertyList props*. Naming contexts bound using this operation participate in name resolution when compound names are resolved.

Parameters

n
The **Name** of the name-object binding.

obj
The naming context object to be bound.

props
The **PropertyList** to associated with the binding.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client continues the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **CosNaming::NamingContext::AlreadyBound** is raised to indicate that an object is already bound to the name. Rebinding operations unbind the name, then rebind the name without raising this exception.
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::ConflictingPropertyName** is raised to indicate that the property name is in conflict.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

bind_with_properties Operation

Creates a name-object binding and associates properties to the binding.

IDL Syntax

void bind_with_properties (
in Name *n*,
in SOMObject *obj*,
in PropertyList *prop*);

Description

Binds an object with properties. Operates just like the **CosNaming::NamingContext::bind** operation in that it binds the specified *SOMObject obj* into the target extended naming context. In addition, it defines properties to be associated with the binding in *PropertyList prop* (combination of **add_properties** and **bind**). A property is replaced if it already exists.

Parameters

n
The **Name** of the name-object binding.

obj
The **SOMObject** to be bound.

prop
The **PropertyList** to associated with the binding.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **CosNaming::NamingContext::AlreadyBound** is raised to indicate that an object is already bound to the name. Rebinding operations unbind the name, then rebind the name without raising this exception.
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::ConflictingPropertyName** is raised to indicate that the property name is in conflict.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

find_all Operation

Retrieves all name-object bindings satisfying property search constraints.

IDL Syntax

void find_all (
in Constraint *c*,
in unsigned long *distance*,
in unsigned long *howMany*,
out BindingList *bl*,
out BindingIterator *bi*);

Description

Outputs each **CosNaming::Binding** that satisfies property search constraint *Constraint c*. It searchs up to a depth of *distance* for all **Bindings** that satisfy the given constraint and puts them into *BindingList bl*. If *distance* is set to 0, this operation searches only the targeted context. Up to *howMany* name-object bindings are placed into the *BindingList bl*. If more than *howMany* objects are found to satisfy the constraint, the remaining name-object bindings are placed into the *BindingIterator bi*.

Parameters

c
The search constraint.

distance
The search depth.

howMany
The maximum number of **Bindings** to put into *bl*.

bl
The outputted **BindingList.**

bi
The outputted **BindingIterator**.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::IllegalConstraintExpression** is raised to indicate that a constraint expression could not be parsed.
- **ExtendedNaming::ExtendedNamingContext::BindingNotFound** is raised to indicate that the search failed.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

find_any Operation

Retrieves the first bound object that satisfies the given search constraint.

IDL Syntax

SOMObject find_any (
in Constraint *c*,
in unsigned long *distance*);

Description

Returns the first bound **SOMObject** satisfying property search constraint *Constraint c*. The returned **SOMObject** contains properties that satisfy *Constraint c*. It searchs up to a depth of *distance* for a binding that satisfies the given constraint. If *distance* is set to 0, this operation searches only the targeted context.

Parameters

c
The search constraint.

distance
The search depth in the Naming Service graph.

Return Value

A **SOMObject** is returned, which satisfies the property search constraint.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate that implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::IllegalConstraintExpression** is raised to indicate that a constraint expression could not be parsed.
- **ExtendedNaming::ExtendedNamingContext::BindingNotFound** is raised to indicate that the search failed.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

find_any_name_binding Operation

Retrieves a name-object binding satisfying property search constraints.

IDL Syntax

void find_any_name_binding (
in Constraint *c*,
in unsigned long *distance*,
out Binding *bi*);

Description

Outputs a **CosNaming::Binding** satisfying property search constraints *Constraint c*. The retrieved **CosName::Binding** is any name-object binding that contains properties that satisfy *Constraint c*. It searches up to a depth of *distance* for a binding that satisfies the given constraint. If *distance* is set to 0, this operation searches only the targeted context.

Parameters

c
The search constraint.

distance
The search depth in the Naming Service graph.

bi
The outputted **Binding**.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::IllegalConstraintExpression** is raised to indicate that a constraint expression could not be parsed.
- **ExtendedNaming::ExtendedNamingContext::BindingNotFound** is raised to indicate that a requested binding was not found.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

get_all_properties Operation

Retrieves all properties for a name-object binding.

IDL Syntax

void et_all_properties (
in Name *n*,
in unsigned long *howMany*,
out PropertyList *props*,
out PropertyIterator *rest*);

Description

Returns all properties for a name-object binding. Returns the properties that are associated with the name-object binding, specified by *Name n*, in the target extended naming context. If the name-object binding contains more than *howMany* properties, then the remaining properties are put in *ExtendedNaming::PropertyIterator rest*. Clients can iterate through the interator to retrieve the remaining properties.

Parameters

n
The **Name** of the name-object binding.

howMany
The maximum number of properties to put into *props*.

props
The returned properties.

rest
The returned **PropertyIterator**.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

Related Information

ExtendedNaming::PropertyIterator Interface

get_features_supported Operation

Retrieves the supported features.

IDL Syntax

unsigned short get_features_supported ( )

Description

Returns the supported features of an extended naming context. Gets a bit vector that this extended naming context implementation supports: 0 properties, 1 shared property, 2 searching, 3 indexing, 4 restrictions on object types, 5 restrictions on property types, 6 restrictions on property names, 7 - 15 not used.

Return Value

An unsigned short bit vector is returned indicating supported features.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

get_properties Operation

Retrieves property values for the specified property name.

IDL Syntax

```
void get_properties (
                        in Name n,
                        in unsigned long howMany,
                        in IList inames,
                        out PropertyList props,
                        out PropertyIterator rest );
```

Description

Returns a set of properties for a name-object binding. Returns the properties, with their property names specified as *ExtendedNaming::IList inames*, associated with the name-object binding specified by *Name n* in the target extended naming context. If the name-object binding contains more than *howMany* properties, the remaining properties are put in *ExtendedNaming::PropertyIterator rest*. Clients can iterate through the interator to retrieve the remaining properties.

Parameters

n
The Name of the name-object binding.

howMany
The maximum number of properties to put in *props*.

inames
The list of property names to be retrieved.

props
The returned properties.

rest
The returned **PropertyIterator**.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.

- **ExtendedNaming::ExtendedNamingContext::PropertyNotFound{CosNaming Istring property_name;};** is raised to indicate that a property was not found.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

Related Information

ExtendedNaming::PropertyIterator Interface

get_property Operation

Retrieves the value of the specified property name.

IDL Syntax

void get_property (
in Name *n*,
in Istring *pn*,
out Property *prop*);

Description

Returns a property (value of the property) for a name-object binding. Returns the property, with its property name specified as *CosNaming::Istring pn*, associated with the name-object binding specified by *Name n* in the target extended naming context.

Parameters

n
The **Name** of the name-object binding.

pn
The property name to be outputted.

prop
The returned property.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::PropertyNotFound{CosNaming Istring property_name;};** is raised to indicate that a property was not found.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

list_indexes Operation

Retrieves all defined indexes.

IDL Syntax

void list_indexes (
in unsigned long *howMany*,
out IndexDescriptorList *il*,
out IndexIterator *rest*);

Description

Returns all indexes defined in the target extended naming context. If any bindings in the target extended naming context have properties that are part of indexes in a parent context, those indexes are not listed. Up to *howMany* indexes are placed into the *ExtendedNaming::IndexDescriptorList il*. If more than *howMany* indexes are found, the remaining indexes are put into the *ExtendedNaming::IndexIterator rest*.

Parameters

howMany
The maximum number of indexes.

il
The returned **IndexDescriptorList.**

rest
The returned **IndexIterator**.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

ExtendedNaming::ExtendedNamingContext::NotSupported is raised to indicate that implementation does not support this operation.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

Related Information

ExtendedNaming::IndexIterator Interface

list_properties Operation

Retrieves all **PropertyBindings** for a name-object binding.

IDL Syntax

void list_properties (
in Name *n*,
in unsigned long *howMany*,
out PropertyBindingList *pbl*,
out PropertyBindingIterator *rest*);

Description

Returns all **PropertyBindings** for a name-object binding. Returns all of the **PropertyBindings** (a structural part of an **ExtendedNaming::Property**) that are associated with a name-object binding specified by *Name n,* in the target extended naming context (including both shared and unshared **PropertyBindings**). If the name-object binding contains more than *howMany* **PropertyBindings**, the remaining **PropertyBindings** are put in *ExtendedNaming::PropertyBindingIterator rest*.

Parameters

n
The **Name** of the name-object binding.

howMany
The maximum number of **PropertyBindings**.

pbl
The returned **PropertyBindingList**.

rest
The returned **PropertyBindingIterator**.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

Related Information

ExtendedNaming::PropertyBindingIterator Interface

rebind_context_with_properties Operation

Re-creates a name-**NamingContext** object binding and associates properties.

IDL Syntax

void rebind_context_with_properties (
in Name ***n*****,**
in ExtendedNamingContext ***obj*****,**
in PropertyList ***props*** **);**

Description

Rebinds a naming context with properties. Operates just like the **CosNaming::NamingContext::rebind_context** operation in that it rebinds the specified naming context into the target extended naming context. In addition, it defines the properties in *PropertyList props* to be associated with the binding. If a property is already associated with the binding, it replaces the existing property with the new property. If the property is not already associated with the binding, a new property is associated. Existing properties associated with the binding that are not specified in *PropertyList props* remain intact. Naming contexts bound using this operation participate in name resolution when compound names are resolved.

Parameters

n
The **Name** of the binding.

obj
The naming context to be bound.

props
The **PropertyList** to associated with the binding.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **InvalidName**, **NotFound**, **InvalidPropertyName**, **NotSupported**, **ConflictingPropertyName**
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate that implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::ConflictingPropertyName** is raised to indicate that the property name is in conflict.
- **ExtendedNaming::ExtendedNamingContext::PropertyNotFound{CosNaming Istring property_name;};** is raised to indicate that a property was not found.
- **ExtendedNaming::ExtendedNamingContext::NonSharableProperties** is raised to indicate that properties were attempted to be shared and are not sharable properties.
- **ExtendedNaming::ExtendedNamingContext::PropertiesNotShared** is raised to indicate that properties were not shared.
- **ExtendedNaming::ExtendedNamingContext::IllegalConstraintExpression** is raised to indicate that a constraint expression could not be parsed.

- **ExtendedNaming::ExtendedNamingContext::BindingNotFound** is raised to indicate that a requested binding was not found.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

rebind_with_properties Operation

Re-creates a name-object binding and associate properties.

IDL Syntax

void rebind_with_properties (
in Name *n*,
in SOMObject *obj*,
in PropertyList *props*);

Description

Rebinds an object with properties. Operates just like the **CosNaming::NamingContext::rebind** in that the specified *SOMObject obj* is rebound into the target extended naming context. In addition, it defines the properties in *PropertyList prop* to be associated with the binding. If a property is already associated with the binding, it replaces the existing property with the new property. If the property is not already associated with the binding, a new property is then associated. Existing properties associated with the binding that are not specified in *PropertyList prop* remain intact.

Parameters

n
The **Name** of the name-object binding for rebinding.

obj
The **SOMObject** to be bound.

props
The **PropertyList** to associated with the binding.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate that implementation does not support this operation.
- **ExtendedNaming::ExtendedNamingContext::ConflictingPropertyName** is raised to indicate that the property name is in conflict.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

remove_all_properties Operation

Removes all properties associated with name-object binding.

IDL Syntax

void remove_all_properties (in Name *n*);

Description

Removes all properties associated with name-object binding. Resolves *Name n* in the target extended naming context and removes all properties associated with the binding. If any property is a shared property, the sharing relationship is destroyed.

Parameters

n

The **Name** of the name-object binding.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate implementation does not support this operation.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

remove_index Operation

Removes a specified index.

IDL Syntax

void remove_index (in IndexDescriptor *i*);

Description

Removes a specified index from the target extended naming context. The *distance* is ignored in the *IndexDescriptor i*.

Parameters

i
The index to be removed.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

remove_properties Operation

Removes a set of properties associated with name-object binding.

IDL Syntax

void remove_properties (
in Name *n*,
in IList *plist*);

Description

Removes a set of properties associated with name-object binding. Resolves *Name n* in the target extended naming context and removes the properties whose property names are specified by *ExtendedNaming::IList plist*. If any properties are shared properties, the sharing relationship is destroyed.

Parameters

n
The Name of the name-object binding.

plist
A list of property names for removal.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;}** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::PropertyNotFound{CosNaming Istring property_name;};** is raised to indicate that a property was not found.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate implementation does not support this operation.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

remove_property Operation

Removes a property associated with name-object binding.

IDL Syntax

void remove_property (
in Name *n*,
in Istring *prop*);

Description

Removes a property associated with name-object binding. Resolves *Name n* in the target extended naming context and removes the property whose property name is specified by *CosNaming::Istring prop*. If the property is a shared property, the sharing relationship is destroyed.

Parameters

n
The **Name** of the name-object binding.

prop
The property name.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::PropertyNotFound{CosNaming Istring property_name;};** is raised to indicate that a property was not found.
- **ExtendedNaming::ExtendedNamingContext::NotSupported** is raised to indicate that implementation does not support this operation.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

resolve_with_all_properties Operation

Resolves a name-object binding (returns an object associated with a name) and obtains all associated properties.

IDL Syntax

SOMObject resolve_with_all_properties (
in Name *n*,
in unsigned long *howMany*,
out PropertyList *props*,
out PropertyIterator *rest*);

Description

Resolves a name-object binding and outputs all associated properties. Operates just like the **CosNaming::NamingContext::resolve** operation in that it resolves the specified name-object binding, specified by *CosNaming::Name n*, in the target extended naming context. In addition, it outputs all properties associated with name-object binding. If the name-object binding contains more than *howMany* properties, the remaining properties are put in *ExtendedNaming::PropertyIterator rest*. This operation is a combination of the **resolve** operation and **get_all_properties** operation.

Parameters

n
The Name of the name-object binding.

howMany
The maximum number of properties to put into *props*.

props
The outputted properties.

rest
The outputted **PropertyIterator**.

Return Value

A **SOMObject** is returned, which is the resolved object.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

Related Information

ExtendedNaming::PropertyIterator Interface

resolve_with_properties Operation

Resolves a name-object binding (returns an object associated with a name) and obtains a set of associated properties.

IDL Syntax

SOMObject resolve_with_properties (
in Name *n*,
in unsigned long *howMany*,
in IList *inames*,
out PropertyList *props*,
out PropertyIterator *rest*);

Description

Resolves a name-object binding and outputs a set of associated properties. Operates just like the **CosNaming::NamingContext::resolve** operation in that it resolves the specified name-object binding, specified by *CosNaming::Name n*, in the target extended naming context. It defines properties to be outputted, with their property names specified as *ExtendedNaming::IList inames.* If the name-object binding contains more than *howMany* properties, the remaining properties are put in *ExtendedNaming::PropertyIterator rest*.

Intended Usage

This operation is typically not overridden.

Parameters

n
The **Name** of the name-object binding.

howMany
The maximum number of properties to put into *props.*

inames
List of property names.

props
The returned properties.

rest
The returned **PropertyIterator.**

Return Value

A **SOMObject** is returned, which is the resolved object.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.

- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::PropertyNotFound{CosNaming Istring property_name;};** is raised to indicate that a property was not found.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

Related Information

ExtendedNaming::PropertyIterator Interface

resolve_with_property Operation

Resolves a name-object binding (returns an object associated with a name) and obtains an associated property value.

IDL Syntax

SOMObject resolve_with_property (
in Name n,
***in* Istring *prop*,**
out any *v*);

Description

Resolves a name-object binding (returns an object associated with a name) and outputs the associated property value. Operates just like the **CosNaming::NamingContext::resolve** operation in that it resolves the specified name-object binding, specified by *CosNaming::Name n*, in the target extended naming context. In addition, it retrieves the value of the property *CosNaming::Istring prop* associated with Name n.

Parameters

n
The **Name** of the name-object binding.

prop
The property name.

v
The outputted property value.

Return Value

A **SOMObject** is returned, which is the resolved object.

Exceptions

CORBA 1.1 standard exceptions and the following user exceptions:

- **CosNaming::NamingContext::NotFound{NotFoundReason why; Name rest_of_name;};** is raised to indicate that the name does not identify a binding. If a compound name is passed as an argument for the **bind** operation, it traverses multiple contexts. A **NotFound** exception is raised if any of the intermediate contexts cannot be resolved.
- **CosNaming::NamingContext::CannotProceed{NamingContext ctx; Name rest_of_name;};** is raised to indicate that the implementation has given up for some reason. The client may be able to continue the operation using the returned naming context.
- **CosNaming::NamingContext::InvalidName** is raised to indicate that the name is invalid. A name with a length of zero is invalid. (This exception may be raised upon further implementation restrictions.)
- **ExtendedNaming::ExtendedNamingContext::InvalidPropertyName** is raised to indicate that the property name is invalid. A property name with a length of zero is invalid.
- **ExtendedNaming::ExtendedNamingContext::PropertyNotFound{CosNaming Istring property_name;};** is raised to indicate that a property was not found.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

_get_allowed_object_types Operation

Retrieves a list of types of objects that can be bound.

IDL Syntax

_IDL_SEQUENCE_TypeCode _get_allowed_object_types ( )

Description

Retrieves a list of types of objects that can be bound into the target extended naming context. An empty list implies no restrictions. This implementation places no restrictions on object types.

Intended Usage

Clients typically use this operation to determine whether the naming context implementation places any restrictions on allowed object types.

Return Value

An **_IDL_SEQUENCE_TypeCode** is returned containing the allowed object types.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

_get_allowed_property_names Operation

Retrieves a list of names of properties that can be added.

IDL Syntax

_IDL_SEQUENCE_string _get_allowed_property_names ( )

Description

Retrieves a list of names of properties that can be added to the target extended naming context. An empty list implies no restrictions.

Return Value

An **_IDL_SEQUENCE_string** is returned indicating the allowed property names.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

_get_allowed_property_types Operation

Retrieves a list of the types of the properties that can be added.

IDL Syntax

_IDL_SEQUENCE_TypeCode _get_allowed_property_types ( )

Description

Retrieves a list of the types of the properties that can be added to the target extended naming context. An empty list implies no restrictions. This implementation places no restrictions on the **type** of the allowed property.

Return Value

An **_IDL_SEQUENCE_TypeCode** is returned indicating the allowed property types.

Exceptions

CORBA 1.1 standard exceptions.

Original Interface

ExtendedNaming::ExtendedNamingContext Interface

Appendix A. BNF for Naming Constraint Language

The Naming Service allows searches based on properties attached to a name object binding. Service providers register their service and use *properties* to describe the service offered. Potential clients can then use a constraint expression to describe the requirements that service providers must satisfy. Constraints are expressed in a constraint language. Using the constraint language, you can specify arbitrarily complex expressions that involve property names and potential values.

The constraint language described below is an excerpt from Appendix B of the *Common Object Services Specification Volume 1* (OMG Document Number 94-1-1). It has been slightly modified to support future enhancements.

```
ConstraintExpr  : Expr
                ;
Expr            :    Expr "or" Expr
                |    Expr "and" Expr
                |    Expr "xor" Expr
                |    '(' Expr ')'
                |    NumExpr Op NumExpr
                |    StrExpr Op StrExpr
                |    NumExpr Op StrExpr
                ;
NumExpr         :    NumExpr "+" NumTerm
                |    NumExpr "-" NumTerm
                |    NumTerm
                ;
NumTerm         :    NumFactor
                |    NumTerm "*" NumFactor
                |    NumTerm "/" NumFactor
                ;
NumFactor       :    Num
                |    Identifier
                |    '(' NumExpr ')'
                |    '-' NumFactor
                ;
StrExpr         :    StrTerm
                |    StrExpr "+" StrTerm
                ;
StrTerm         :    String
                |    '(' StrExpr ')'
                ;
                ;
Op              :    "==" | "<=" | ">=" | "!=" | "<" | ">"
                ;
Identifier      :    Word
                ;
Word            :    Letter { AlphaNum }+
                ;
AlphaNum        :    Letter
                |    Digit
                |    "_"
                ;
String          :    "'" { Char }* "'"
                ;
Num             :    { Digit}+
                |    { Digit}+ "." { Digit}*
                ;
Char            :    Letter
                |    Digit
                |    Other
                ;
```

```
Letter          :    a | b | c | d | e | f | g | h | i
                |    j | k | l | m | n | o | p | q | r
                |    s | t | u | v | w | x | y | z | A
                |    B | C | D | E | F | G | H | I | J
                |    K | L | M | N | O | P | Q | R | S
                |    T | U | V | W | X | Z
                ;
Digit           :    0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
                ;
Other           :    <Sp> | ~ | ! | @ | # | $ | % | ^ | &
                |    * | ( | ) | - | _ | = | + | [ | {
                |    ] | } | ; | : | " | \ | | | , | <
                |    . | > | / | ?
                ;
Sp              :    " "
                ;
```

The following precedence relations hold in the absence of parentheses, in the order of lowest to highest:

- *or* and *xor*
- *and*
- *not*
- + and -
- * and /
- Otherwise, left-to-right precedence

The following are some example constraints:

```
(1) name == 'ashoo'
(2) name == 'ashoo' and pet == 'flakes'
(3) Fee <= 5 or LowFreq >= 20
(4) DeviceType == 'Car' and Cost < 30000 and color == 'white'
     and Year > 1990
```

Index

R

Printed in U.S.A.

.