Oracle CDD/Repository™

Information Model
Volume I

Version 5.0

Reprinted 1995

Part No. A24847-2

ORACLE®

Information Model
Volume I

Version 5.0

Part Number A24847-2

Contents

2 Message Descriptions

3 Property Descriptions

Tables

Send Us Your Comments

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

You can send comments to us in the following ways:

- Electronic mail — nedc_doc@us.oracle.com
- FAX — 603-897-3334 Attn: Oracle CDD/Repository Documentation
- Postal service

  Oracle Corporation
  Oracle CDD/Repository Documentation
  One Oracle Drive
  Nashua, NH 03062
  USA

If you like, you can use the following questionnaire to give us feedback.

Name _____ Title _____

Company _____ Department _____

Mailing Address _____ Telephone Number _____

Book Title _____ Version Number _____

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?

- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available).

Preface

This manual contains the definition of the fundamental element types, their properties, the messages that can be sent to them, and the methods that respond to those messages.

Intended Audience

This manual is for anyone who integrates applications with Oracle CDD/Repository. Integrated applications include:

- Any enterprise-wide application that requires a common dictionary of data definitions.
- Computer-aided software engineering (CASE) tools that use Oracle CDD/Repository as a repository for software development project data and as a medium for interoperation.
- Other enterprise-wide applications including:
  - Data administration and system architecture planning—Oracle CDD/Repository stores business models and data, and supports the integration of modeling tools.
  - End-user development—Oracle CDD/Repository stores user data, controls development process, enforces corporate rules and standards, and tracks users of corporate data.
  - Extracted file access—Oracle CDD/Repository tracks and manages file creation, and provides extract dates and data definitions.

Use this manual for reference throughout the development process.

Document Structure

This manual contains the following chapters and appendixes:

- Chapter 1 contains summary information about each element type.
- Chapter 2 contains descriptions of the Oracle CDD/Repository messages, their arguments, and the methods that implement them.
- Chapter 3 contains descriptions of the Oracle CDD/Repository properties.
- Appendix A contains summary information about the relation properties, organized by relation type and by property.
- Appendix B contains implemented information model (IIM) diagrams for many of the element types and relation types described in this manual.

Related Documents

Documents related to Oracle CDD/Repository include the following:

- *Oracle CDD/Repository CDO Reference Manual*
- *Using Oracle CDD/Repository on OpenVMS Systems*
- *Oracle CDD/Repository Architecture Manual*
- *Oracle CDD/Repository Callable Interface Manual*
- *Oracle CDD/Repository Information Model Volume I*
- *Oracle CDD/Repository Information Model Volume II*
- *Installing Oracle CDD/Repository on OpenVMS Systems*
- *Read Before Installing or Using Oracle CDD/Repository on OpenVMS VAX Systems* or, depending on your system, *Read Before Installing or Using Oracle CDD/Repository on OpenVMS Alpha Systems*

See online help for a glossary of defined terms.

Conventions

Table 1 shows the conventions used in this manual.

Table 1 Documentation Conventions

Convention	Meaning
italic type	Italic type emphasizes important information, indicates variables, and indicates complete titles of manuals.
UPPERCASE	Words in uppercase indicate a command, the name of a file, the name of a file protection code, or an abbreviation for a system privilege.
NAMED_ELEMENT	The names of element types are set in small capitals.
merge **hasChildren**	The names of elements, messages, and properties are set in bold type.
list_value	References to values you supply, such as arguments, are set in italics.
0 ELEMENT 1 NAMED_ELEMENT 2 VERSION	The supertype–subtype relationship between element types is represented by numbers and by indentation. The example indicates that NAMED_ELEMENT is a supertype of VERSION and a subtype of ELEMENT.

1
Element Type Descriptions

This chapter contains descriptions of the element types provided with Oracle CDD/Repository, arranged alphabetically. Each element description contains the following sections:

Title—Includes the generic name of the element type, a short phrase that describes the type, and a paragraph outlining its purpose.

Symbolic Name—Gives the symbolic names of the element type in the C and OpenVMS bindings. Use these symbols when a routine requires you to identify an element type by name.

Hierarchy Level—Gives the position of the element type in the type hierarchy. This illustrates the type's supertypes up to the root of the hierarchy. Use this diagram to determine the types from which properties and methods are inherited.

Figure 1–1 illustrates the complete element type hierarchy.

Tag—Gives the symbolic constant for the value of the **tag** property for this element type or relation type.

Defined Properties—Lists properties defined by this element type. Refer to Chapter 3 for a description of each property. If you must supply a value for a property when you create new instances of the element type, the name of the property is followed by one of these legends:

- *Required*—You must always supply a value for the property.
- *Conditionally required*—You must supply a value for the property under certain circumstances. See the description of the **new** message as it applies to this element type in Chapter 2 for more information.

Inherited Properties—Lists properties inherited by this element type. The name of the element type that defines the property appears in parentheses. Refer to Chapter 3 for a description of each property.

Defined Methods—Lists messages for which this type provides a method and for which none of its supertypes (direct or indirect) provide a method. A description of the method appears with the description of the message in Chapter 2.

Refined Methods—Lists messages for which this type provides a method and for which one or more supertypes also provide a method. The name of the nearest supertype to provide a method appears in parentheses. Generally, a refined method does some type-specific processing before and/or after calling the supertype's method. A description of the method appears with the description of the message in Chapter 2.

Inherited Methods—Lists messages for which this type does *not* provide a method and for which one or more supertypes do provide a method. The name of the nearest supertype to provide a method appears in parentheses.

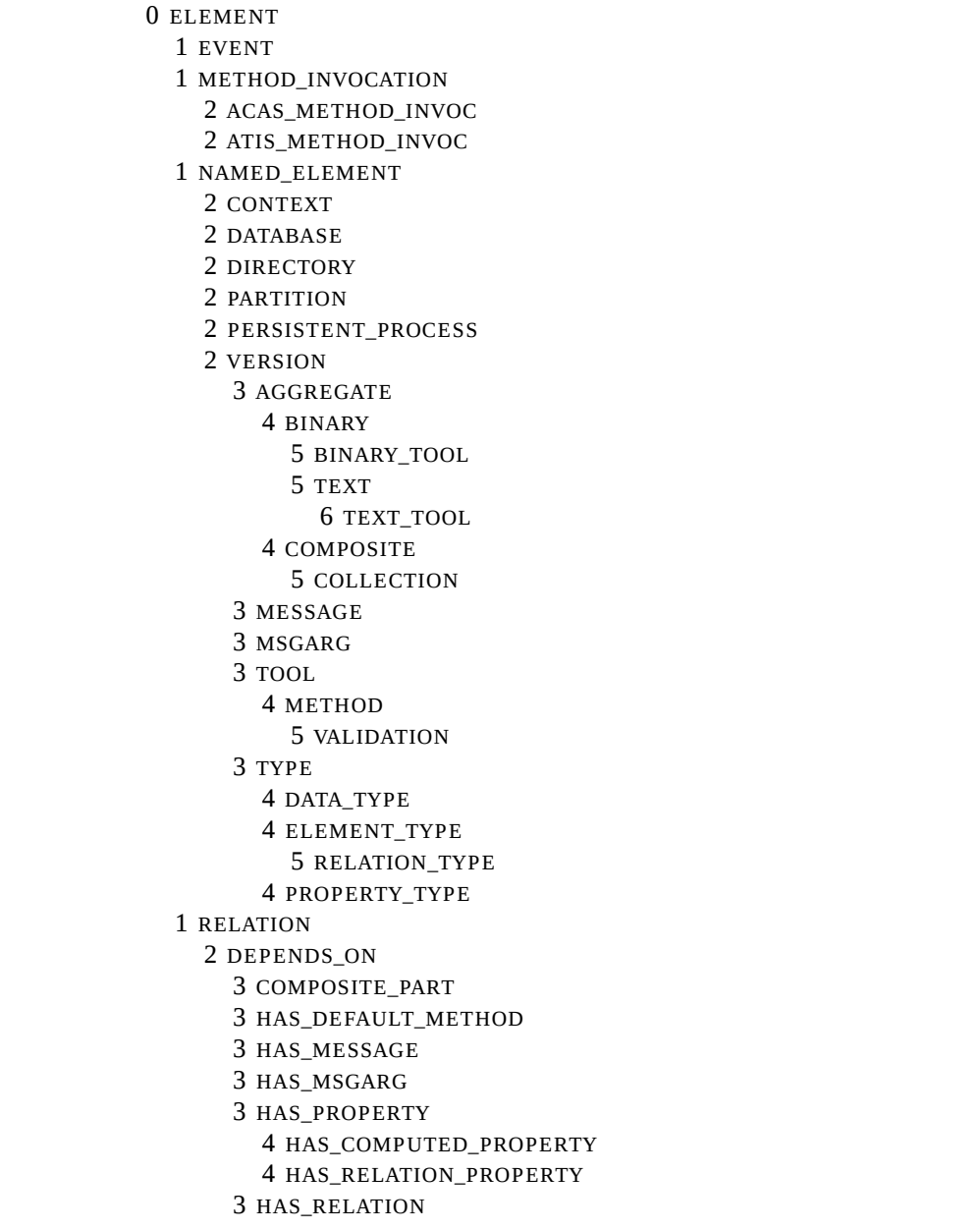
Methods Not Allowed—Lists messages for which one or more supertypes provide a method but which this type specifically disallows. The message description in Chapter 2 states why the method is not allowed for this type.

The Participants section applies only to subtypes of RELATION. It describes the characteristics for the owner and member participants of the defined relationship. These characteristics are the following:

- **Legal Values**—One or more element types that can participate in the relationship. Subtypes of the type(s) can also participate.
- **Many**—If TRUE, the participant can participate in other relationships of this type in the same capacity (that is, as owner or member). (This information is provided to help clarify the metadata schema. This release of Oracle CDD/Repository has no direct way of restricting the number of relationships of a given type in which an element can participate.)

By looking at the hierarchy level, note whether the relation is a subtype of DEPENDS_ON. If so, the relation defines dependency relationships.

Figure 1–1 Element Type Hierarchy Inheritance Summary



(continued on next page)

Figure 1–1 (Cont.) Element Type Hierarchy Inheritance Summary

```
      3 HAS_SUPERTYPE
      3 IMPLEMENTS_METHOD
      3 IMPLEMENTS_RELATION
      3 INVOKES_TOOL
      3 RELATION_MEMBER
   2 HAS_CONTEXT
   2 HAS_CURR_COLLECTION
   2 HAS_DATATYPE
   2 HAS_PARENT
   2 HAS_POSTAMBLE
   2 HAS_PREAMBLE
   2 HAS_RELATED_PARTITION
   2 HAS_TOP_COLLECTION
   2 OPENED_BY
   2 RESERVED_BY
```

ACAS_METHOD_INVOC—ACA Services Method Invocation

Defines elements that represent the invocation of ACA Services methods to build derived objects from source objects. These elements record dependencies between inputs, outputs, and processors. This element type is a specialization of the METHOD_INVOCATION type that records the results of sending ACA Services messages, as opposed to Oracle CDD/Repository messages.

Summary

Symbolic Name

C binding	MCS_elm_acas_method_invoc
OpenVMS binding	MCS$r_elm_acas_method_invoc
Hierarchy Level	0 ELEMENT 1 METHOD_INVOCATION 2 ACAS_METHOD_INVOC
Tag	NAD$K_ENT_ACAS_MI

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
CPUTime (METHOD_INVOCATION)
createdDate (METHOD_INVOCATION)
derivedFrom (METHOD_INVOCATION)
derives—*Required* (METHOD_INVOCATION)
elapsedTime (METHOD_INVOCATION)
elementType (ELEMENT)
invocationStatus (METHOD_INVOCATION)
invocationString (METHOD_INVOCATION)
logFile (METHOD_INVOCATION)
optionsString (METHOD_INVOCATION)
OSVersion (METHOD_INVOCATION)
scalingFactor (METHOD_INVOCATION)
toolVersion (METHOD_INVOCATION)

Defined Methods

-none-

ACAS_METHOD_INVOC

Refined Methods

-none-

Inherited Methods

free (ELEMENT)
getProp (ELEMENT)
new (ELEMENT)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

AGGREGATE—Aggregate

Defines compound elements that contain other elements as subparts. This element type exists for the purpose of defining methods that are refined by its subtypes.

Summary

Symbolic Name	
C binding	MCS_elm_aggregate
OpenVMS binding	MCS$r_elm_aggregate
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 AGGREGATE
Tag	NAD$K_REL_AGGREGATE

Defined Properties

-none-

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)

AGGREGATE

hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

close
export
import
open

Refined Methods

-none-

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
free (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
new (VERSION)
promote (VERSION)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (VERSION)
reserve (VERSION)
setProp (VERSION)
translate (VERSION)

unfreeze (VERSION)
unreserve (VERSION)
verify (ELEMENT)

Methods Not Allowed

-none-

ATIS_METHOD_INVOC—ATIS Method Invocation

Defines elements that represent the invocation of Oracle CDD/Repository methods to build derived objects from source objects. These elements record dependencies between inputs, outputs, and processors. This element type is a specialization of the METHOD_INVOCATION type that records the results of sending Oracle CDD/Repository messages, as opposed to ACA Services messages.

Summary

Symbolic Name	
C binding	MCS_elm_atis_method_invoc
OpenVMS binding	MCS$r_elm_atis_method_invoc
Hierarchy Level	0 ELEMENT 1 METHOD_INVOCATION 2 ATIS_METHOD_INVOC
Tag	NAD$K_ENT_ATIS_MI

Defined Properties

argsSent
methodUsed—*Required*
msgSent—*Required*
msgTarget—*Required*

Inherited Properties

allElementTypes (ELEMENT)
CPUTime (METHOD_INVOCATION)
createdDate (METHOD_INVOCATION)
derivedFrom (METHOD_INVOCATION)
derives—*Required* (METHOD_INVOCATION)
elapsedTime (METHOD_INVOCATION)
elementType (ELEMENT)
invocationStatus (METHOD_INVOCATION)
invocationString (METHOD_INVOCATION)
logFile (METHOD_INVOCATION)
optionsString (METHOD_INVOCATION)
OSVersion (METHOD_INVOCATION)
scalingFactor (METHOD_INVOCATION)
toolVersion (METHOD_INVOCATION)

ATIS_METHOD_INVOC

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (ELEMENT)
getProp (ELEMENT)
new (ELEMENT)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

BINARY—General File

Defines the most general elements that represent files. BINARY's defined properties and refined methods implement the Oracle CDD/Repository file system. Subtypes of BINARY define more specific types of files.

BINARY elements whose storage type is internal cannot be uncontrolled.

Summary

Symbolic Name	
C binding	MCS_elm_binary
OpenVMS binding	MCS$r_elm_binary
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 AGGREGATE 4 BINARY
Tag	NAD$K_ENT_BINARY

Defined Properties

deltaFile
filePath
importedFrom
openedBy
referenceCount
storedIn—*Conditionally required*
storeType—*Required*

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)

dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

close (AGGREGATE)
detach (VERSION)
export (AGGREGATE)
free (VERSION)
import (AGGREGATE)
new (VERSION)
open (AGGREGATE)
promote (VERSION)
replace (VERSION)
reserve (VERSION)
unreserve (VERSION)
verify (ELEMENT)

Inherited Methods

attach (VERSION)

BINARY

build (VERSION)
control (VERSION)
edit (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
purge (VERSION)
rename (NAMED_ELEMENT)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)

Methods Not Allowed

-none-

BINARY_TOOL—Binary Tool

Defines elements to represent units of external code that implement one or more methods. These units of code may be either programs or shareable image libraries.

Summary

Symbolic Name	
C binding	MCS_elm_binary_tool
OpenVMS binding	MCS$r_elm_binary_tool
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 AGGREGATE 4 BINARY 5 BINARY_TOOL
Tag	NAD$K_ENT_BINARY_TOOL

Defined Properties

-none-

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
deltaFile (BINARY)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)

BINARY_TOOL

elementType (ELEMENT)
filePath (BINARY)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
importedFrom (BINARY)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
openedBy (BINARY)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
referenceCount (BINARY)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
storedIn—*Conditionally required* (BINARY)
storeType—*Required* (BINARY)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

attach (VERSION)
build (VERSION)
close (BINARY)
control (VERSION)
detach (BINARY)
edit (VERSION)
export (BINARY)
free (BINARY)
freeze (VERSION)
getProp (ELEMENT)
import (BINARY)

merge (VERSION)
new (BINARY)
open (BINARY)
promote (BINARY)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (BINARY)
reserve (BINARY)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
unreserve (BINARY)
verify (BINARY)

Methods Not Allowed

-none-

COLLECTION—Collection

Defines elements that represent system configurations. COLLECTION is a subtype of COMPOSITE. The following are the important differences between collections and composites:

- The graph formed by a collection's children and their children may not contain cycles. A composite has no such restriction.
- COLLECTION elements cannot be uncontrolled.

Summary

Symbolic Name	
C binding	MCS_elm_collection
OpenVMS binding	MCS$r_elm_collection
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 AGGREGATE 4 COMPOSITE 5 COLLECTION
Tag	NAD$K_ENT_COLLECTION

Defined Properties

-none-

Inherited Properties

access (NAMED_ELEMENT)
allChildren (COMPOSITE)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)

dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
hasChildren (COMPOSITE)
hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
numChildren (COMPOSITE)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

attach (VERSION)
close (AGGREGATE)
detach (VERSION)
export (AGGREGATE)
free (VERSION)
import (AGGREGATE)
open (AGGREGATE)
promote (VERSION)
replace (VERSION)
reserve (VERSION)
unreserve (VERSION)

Inherited Methods

build (VERSION)

COLLECTION

control (VERSION)
edit (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (COMPOSITE)
new (VERSION)
purge (VERSION)
rename (NAMED_ELEMENT)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
update (COMPOSITE)
verify (ELEMENT)

Methods Not Allowed

-none-

COMPOSITE—Composite

Defines elements that are logical groups of attached VERSION elements. The **attach** message makes a version part of a composite by creating a COMPOSITE_PART relationship between the COMPOSITE element and the version; the **detach** message removes a version from a composite by deleting the relationship. Attached version members are called children; the composite to which they are attached is called their parent.

COMPOSITE elements are under version control. This means that a succession of versions of a composite can capture changing groups of versions. Thus, composites can represent the evolution of a group of versions in the same way that versions represent the evolution of a single object.

Summary

Symbolic Name	
C binding	MCS_elm_composite
OpenVMS binding	MCS$r_elm_composite
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 AGGREGATE 4 COMPOSITE
Tag	NAD$K_ENT_COMPOSITE

Defined Properties

allChildren
hasChildren
numChildren

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)

COMPOSITE

controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

update

Refined Methods

merge (VERSION)

Inherited Methods

attach (VERSION)
build (VERSION)
close (AGGREGATE)
control (VERSION)
detach (VERSION)
edit (VERSION)
export (AGGREGATE)
free (VERSION)
freeze (VERSION)
getProp (ELEMENT)

import (AGGREGATE)
new (VERSION)
open (AGGREGATE)
promote (VERSION)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (VERSION)
reserve (VERSION)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
unreserve (VERSION)
verify (ELEMENT)

Methods Not Allowed

-none-

COMPOSITE_PART—Composite Part

Defines relationships that connect COMPOSITE elements to the members of the composite. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_composite_part
OpenVMS binding	MCS$r_elm_composite_part

Hierarchy Level

0 ELEMENT
1 RELATION
2 DEPENDS_ON
3 COMPOSITE_PART

Tag: NAD$K_REL_COMPOSITE_PART

Defined Properties

attachment

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	COMPOSITE	true
member	VERSION	true

CONTEXT—Context

Defines elements that represent views of the repository. CONTEXT elements identify a collection that represents a configuration, a partition that is the base of the partition hierarchy, and a file system directory that contains configuration files in subdirectories.

Summary

Symbolic Name	
C binding	MCS_elm_context
OpenVMS binding	MCS$r_elm_context
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 CONTEXT
Tag	NAD$K_ENT_CONTEXT

Defined Properties

basePartition—*Required*
checkout
contextDir
defaultAccess
defaultAttachment
openedFiles
top

Inherited Properties

access (NAMED_ELEMENT)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
createdDate (NAMED_ELEMENT)
description (NAMED_ELEMENT)
elementType (ELEMENT)
history (NAMED_ELEMENT)
name (NAMED_ELEMENT)—*Required*
owner (NAMED_ELEMENT)
processingName (NAMED_ELEMENT)
simpleName (NAMED_ELEMENT)

Defined Methods

close

open

Refined Methods

free (ELEMENT)
new (NAMED_ELEMENT)
setProp (ELEMENT)
verify (ELEMENT)

Inherited Methods

getProp (ELEMENT)
rename (NAMED_ELEMENT)

Methods Not Allowed

-none-

DATABASE—Database

Defines elements that represent Oracle CDD/Repository databases. Each repository contains a DATABASE element that describes the containing repository as well as one for each repository that is known to the repository. The DATABASE element that represents the containing repository is in the CDD$PROTOCOLS directory and is named CDD$SELF.

Summary

Symbolic Name	
C binding	MCS_elm_database
OpenVMS binding	MCS$r_elm_database
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 DATABASE
Tag	NAD$K_ENT_ANCHOR

Defined Properties

allCheckouts
defaultAccess
node—*Required*
path—*Required*
rootPath

Inherited Properties

access (NAMED_ELEMENT)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
createdDate (NAMED_ELEMENT)
description (NAMED_ELEMENT)
elementType (ELEMENT)
history (NAMED_ELEMENT)
name (NAMED_ELEMENT)
owner (NAMED_ELEMENT)
processingName (NAMED_ELEMENT)
simpleName (NAMED_ELEMENT)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

getProp (ELEMENT)
rename (NAMED_ELEMENT)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

free
new

DATA_TYPE—Data Type

Defines elements that represent the basic Oracle CDD/Repository data types. Users cannot create DATA_TYPE elements to create new data types.

Summary

Symbolic Name	
C binding	MCS_elm_data_type
OpenVMS binding	MCS$r_elm_data_type
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 TYPE 4 DATA_TYPE
Tag	NAD$K_ENT_DATA_TYPE

Defined Properties

baseType
baseTypeSize

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
allSubTypes (TYPE)
allSuperTypes (TYPE)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)

elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
subTypes (TYPE)
superTypes (TYPE)
tag (TYPE)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
promote (TYPE)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (TYPE)
reserve (TYPE)
setProp (VERSION)

DATA_TYPE

translate (VERSION)
unfreeze (VERSION)
verify (ELEMENT)

Methods Not Allowed

new

DEPENDS_ON—Dependency Relation

Defines dependency relationships. Dependency relationships are owned by elements that should be notified if a relationship member changes. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_depends_on
OpenVMS binding	MCS$r_elm_depends_on
Hierarchy Level	0 ELEMENT 1 RELATION 2 DEPENDS_ON
Tag	NAD$K_REL_DEPENDS

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

setProp (ELEMENT)

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
verify (ELEMENT)

Methods Not Allowed

-none-

DEPENDS_ON

Participants

Participant	Legal Values	Many
owner	ELEMENT	true
member	ELEMENT	true

DIRECTORY—Directory

Defines elements that represent Oracle CDD/Repository directories.

Summary

Symbolic Name
- C binding: MCS_elm_directory
- OpenVMS binding: MCS$r_elm_directory

Hierarchy Level:
0 ELEMENT
1 NAMED_ELEMENT
2 DIRECTORY

Tag: NAD$K_ENT_DIRECTORY

Defined Properties

-none-

Inherited Properties

access (NAMED_ELEMENT)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
createdDate (NAMED_ELEMENT)
description (NAMED_ELEMENT)
elementType (ELEMENT)
history (NAMED_ELEMENT)
name (NAMED_ELEMENT)—*Required*
owner (NAMED_ELEMENT)
processingName (NAMED_ELEMENT)
simpleName (NAMED_ELEMENT)

Defined Methods

-none-

Refined Methods

free (ELEMENT)
new (NAMED_ELEMENT)

Inherited Methods

getProp (ELEMENT)
setProp (ELEMENT)
verify (ELEMENT)

DIRECTORY

Methods Not Allowed

rename

ELEMENT—Element

Defines the most general elements in the repository. ELEMENT is the top of the type hierarchy. It represents the basic instantiable objects in the system and defines properties common to all such objects.

Summary

Symbolic Name	
C binding	MCS_elm_element
OpenVMS binding	MCS$r_elm_element
Hierarchy Level	0 ELEMENT
Tag	NAD$K_ENT_ELEMENT

Defined Properties

allElementTypes
elementType

Inherited Properties

-none-

Defined Methods

free
getProp
new
setProp
verify

Refined Methods

-none-

Inherited Methods

-none-

Methods Not Allowed

-none-

ELEMENT_TYPE—Element Type

Defines elements that represent element types. Each element type in the type hierarchy (including ELEMENT_TYPE itself) is an instance of this type or one of its subtypes.

Summary

Symbolic Name	
C binding	MCS_elm_element_type
OpenVMS binding	MCS$r_elm_element_type
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 TYPE 4 ELEMENT_TYPE
Tag	NAD$K_ENT_OBJECT_TYPE

Defined Properties

allInstances
associatedValidations
compPropDef
definedMethods
definedPropDefs
instances
methods
ownsRelation
pattern
propDef
relationMember
relPropDef
rootBranchInstances
versionable

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)

allSubTypes (TYPE)
allSuperTypes (TYPE)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
subTypes (TYPE)
superTypes (TYPE)—*Required*
tag (TYPE)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

new (VERSION)

Inherited Methods

attach (VERSION)

ELEMENT_TYPE

build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
promote (TYPE)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (TYPE)
reserve (TYPE)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
verify (ELEMENT)

Methods Not Allowed

free

EVENT—History Record

Defines elements that are history records. EVENT elements are created for an element when it is created with the **new** method and each time a message is sent to the element, if the METHOD element that implements the message has its **keepHist** property set to TRUE.

Summary

Symbolic Name	
C binding	MCS_elm_event
OpenVMS binding	MCS$r_elm_event
Hierarchy Level	0 ELEMENT 1 EVENT
Tag	NAD$K_ENT_HISTORY

Defined Properties

argList
contextName
createdDate
historyComment
messageName—*Required*
toolName
userName—*Required*

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)

Defined Methods

-none-

Refined Methods

new (ELEMENT)

Inherited Methods

free (ELEMENT)
getProp (ELEMENT)
setProp (ELEMENT)
verify (ELEMENT)

EVENT

Methods Not Allowed

writeHistory

HAS_COMPUTED_PROPERTY—Has Computed Property

Defines relationships that connect ELEMENT_TYPE elements to PROPERTY_TYPE elements when the property is computed. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_computed_property
OpenVMS binding	MCS$r_elm_has_computed_property

Hierarchy Level

```
0 ELEMENT
  1 RELATION
    2 DEPENDS_ON
      3 HAS_PROPERTY
        4 HAS_COMPUTED_PROPERTY
```

Tag: NAD$K_REL_HAS_COMP_PROP

Defined Properties

implementsMethod—*Required*

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
mutable (HAS_PROPERTY)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)
required—*Required* (HAS_PROPERTY)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

HAS_COMPUTED_PROPERTY

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	ELEMENT_TYPE	true
member	PROPERTY_TYPE	true

HAS_CONTEXT—Has Context

Defines relationships that associate a persistent process with its currently open context. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_context
OpenVMS binding	MCS$r_elm_has_context
Hierarchy Level	0 ELEMENT 1 RELATION 2 HAS_CONTEXT
Tag	NAD$K_REL_HAS_CONTEXT

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_CONTEXT

Participants

Participant	Legal Values	Many
owner	PERSISTENT_PROCESS	false
member	CONTEXT	true

HAS_CURR_COLLECTION—Has Current Collection

Defines relationships that associate a persistent process with its current collection. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_curr_collection
OpenVMS binding	MCS$r_elm_has_curr_collection
Hierarchy Level	0 ELEMENT 1 RELATION 2 HAS_CURR_COLLECTION
Tag	NAD$K_REL_HAS_CURR_COLL

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_CURR_COLLECTION

Participants

Participant	Legal Values	Many
owner	PERSISTENT_PROCESS	false
member	COMPOSITE	true

HAS_DATATYPE—Has Data Type

Defines relationships that associate a property or message argument definition with a data type. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_datatype
OpenVMS binding	MCS$r_elm_has_datatype
Hierarchy Level	0 ELEMENT 1 RELATION 2 HAS_DATATYPE
Tag	NAD$K_REL_HAS_DATATYPE

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_DATATYPE

Participants

Participant	Legal Values	Many
owner	PROPERTY_TYPE MSGARG	false
member	DATA_TYPE	true

HAS_DEFAULT_METHOD—Has Default Method

Defines relationships that associate an element type with the methods that implement messages for the type. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_default_method
OpenVMS binding	MCS$r_elm_has_default_method

Hierarchy Level

0 ELEMENT
1 RELATION
2 DEPENDS_ON
3 HAS_DEFAULT_METHOD

Tag: NAD$K_REL_HAS_DEF_METHOD

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_DEFAULT_METHOD

Participants

Participant	Legal Values	Many
owner	ELEMENT_TYPE	true
member	METHOD	false

HAS_MESSAGE—Has Message

Defines relationships that associate a METHOD element with the message (MESSAGE element) that it implements. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_message
OpenVMS binding	MCS$r_elm_has_message
Hierarchy Level	0 ELEMENT 1 RELATION 2 DEPENDS_ON 3 HAS_MESSAGE
Tag	NAD$K_REL_HAS_MESSAGE

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_MESSAGE

Participants

Participant	Legal Values	Many
owner	METHOD	false
member	MESSAGE	true

HAS_MSGARG—Has Message Argument

Defines relationships that connect MESSAGE elements to the MSGARG elements that define the message's arguments. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_msgarg
OpenVMS binding	MCS$r_elm_has_msgarg

Hierarchy Level

0 ELEMENT
1 RELATION
2 DEPENDS_ON
3 HAS_MSGARG

Tag: NAD$K_REL_HAS_MSGARG

Defined Properties

passingMechanism
required

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

HAS_MSGARG

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	MESSAGE	true
member	MSGARG	true

HAS_PARENT—Has Parent Partition

Defines relations that associate parent and child partitions in the partition hierarchy. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_parent
OpenVMS binding	MCS$r_elm_has_parent
Hierarchy Level	0 ELEMENT 1 RELATION 2 HAS_PARENT
Tag	NAD$K_REL_PARENT_P

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_PARENT

Participants

Participant	Legal Values	Many
owner	PARTITION CONTEXT	false
member	PARTITION	true

HAS_POSTAMBLE—Has Postamble

Defines relationships that associate a METHOD element with the METHOD elements that make up its postamble. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name
- C binding: MCS_elm_has_postamble
- OpenVMS binding: MCS$r_elm_has_postamble

Hierarchy Level:
0 ELEMENT
1 RELATION
2 HAS_POSTAMBLE

Tag: NAD$K_REL_POSTAMBLE

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_POSTAMBLE

Participants

Participant	Legal Values	Many
owner	METHOD	true
member	METHOD	true

HAS_PREAMBLE—Has Preamble

Defines relationships that associate a METHOD element with the METHOD elements that make up its preamble. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name
- C binding: MCS_elm_has_preamble
- OpenVMS binding: MCS$r_elm_has_preamble

Hierarchy Level:
0 ELEMENT
1 RELATION
2 HAS_PREAMBLE

Tag: NAD$K_REL_PREAMBLE

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_PREAMBLE

Participants

Participant	Legal Values	Many
owner	METHOD	true
member	METHOD	true

HAS_PROPERTY—Has Property

Defines relationships that connect ELEMENT_TYPE elements to PROPERTY_TYPE elements. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_property
OpenVMS binding	MCS$r_elm_has_property

Hierarchy Level

0 ELEMENT
  1 RELATION
    2 DEPENDS_ON
      3 HAS_PROPERTY

Tag: NAD$K_REL_HAS_ATT

Defined Properties

mutable
required—*Required*

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

HAS_PROPERTY

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	ELEMENT_TYPE	true
member	PROPERTY_TYPE	true

HAS_RELATED_PARTITION—Has Related Partition

Defines relationships that associate a partition with a related partition. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name
- C binding: MCS_elm_has_related_partition
- OpenVMS binding: MCS$r_elm_has_related_partition

Hierarchy Level:
0 ELEMENT
1 RELATION
2 HAS_RELATED_PARTITION

Tag: NAD$K_REL_LOOKASIDE_P

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_RELATED_PARTITION

Participants

Participant	Legal Values	Many
owner	PARTITION	true
member	PARTITION	true

HAS_RELATION—Has Relation

Defines relationships that connect ELEMENT_TYPE elements to RELATION_TYPE elements. A HAS_RELATION relationship indicates that the elements of the element type may be owners of relationships of the relation type. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name	
C binding	MCS_elm_has_relation
OpenVMS binding	MCS$r_elm_has_relation
Hierarchy Level	0 ELEMENT 1 RELATION 2 DEPENDS_ON 3 HAS_RELATION
Tag	NAD$K_REL_OWNS_REL

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

HAS_RELATION

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	ELEMENT_TYPE	true
member	RELATION_TYPE	true

HAS_RELATION_PROPERTY—Has Relation Property

Defines relationships that connect ELEMENT_TYPE elements to PROPERTY_TYPE elements when the property is implemented by a relation. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_has_relation_property
OpenVMS binding	MCS$r_elm_has_relation_property

Hierarchy Level

```
0 ELEMENT
  1 RELATION
    2 DEPENDS_ON
      3 HAS_PROPERTY
        4 HAS_RELATION_PROPERTY
```

Tag: NAD$K_REL_HAS_REL_PROP

Defined Properties

direction—*Required*
implementsRelation—*Required*

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
mutable (HAS_PROPERTY)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)
required—*Required* (HAS_PROPERTY)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)

HAS_RELATION_PROPERTY

verify (ELEMENT)

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	ELEMENT_TYPE	true
member	PROPERTY_TYPE	true

HAS_SUPERTYPE—Has Supertype

Defines relationships that associate an ELEMENT_TYPE element with its supertype. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name
- C binding: MCS_elm_has_supertype
- OpenVMS binding: MCS$r_elm_has_supertype

Hierarchy Level:
0 ELEMENT
1 RELATION
2 DEPENDS_ON
3 HAS_SUPERTYPE

Tag: NAD$K_REL_HAS_SUPERTYPE

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_SUPERTYPE

Participants

Participant	Legal Values	Many
owner	TYPE	true
member	TYPE	true

HAS_TOP_COLLECTION—Has Top Collection

Defines relationships that associate a CONTEXT element with the version (usually a COLLECTION element) that is the top of the context's collection hierarchy. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name
- C binding: MCS_elm_has_top_collection
- OpenVMS binding: MCS$r_elm_has_top_collection

Hierarchy Level:
0 ELEMENT
  1 RELATION
    2 HAS_TOP_COLLECTION

Tag: NAD$K_REL_TOP

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

HAS_TOP_COLLECTION

Participants

Participant	Legal Values	Many
owner	CONTEXT	false
member	VERSION	true

IMPLEMENTS_METHOD—Method Implements Computed Property

Defines relationships that associate a HAS_COMPUTED_PROPERTY with the METHOD element that computes the computed property's value. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_implements_method
OpenVMS binding	MCS$r_elm_implements_method

Hierarchy Level

0 ELEMENT
1 RELATION
2 DEPENDS_ON
3 IMPLEMENTS_METHOD

Tag: NAD$K_REL_IMP_METHOD

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

Methods Not Allowed

-none-

IMPLEMENTS_METHOD

Participants

Participant	Legal Values	Many
owner	HAS_COMPUTED_PROPERTY	false
member	METHOD	true

IMPLEMENTS_RELATION—Implements Relation

Defines relationships that connect HAS_RELATION_PROPERTY elements to RELATION_TYPE elements. The relationship indicates the relation type that implements a relation property. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name	
C binding	MCS_elm_implements_relation
OpenVMS binding	MCS$r_elm_implements_relation
Hierarchy Level	0 ELEMENT 1 RELATION 2 DEPENDS_ON 3 IMPLEMENTS_RELATION
Tag	NAD$K_REL_IMP_RELATION

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

IMPLEMENTS_RELATION

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	HAS_RELATION_PROPERTY	true
member	RELATION_TYPE	true

INVOKES_TOOL—Invokes Tool

Defines relationships that associate a TOOL element with the BINARY_TOOL or TEXT_TOOL element representing the actual software tool. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_invokes_tool
OpenVMS binding	MCS$r_elm_invokes_tool

Hierarchy Level

0 ELEMENT
  1 RELATION
    2 DEPENDS_ON
      3 INVOKES_TOOL

Tag: NAD$K_REL_INVOKES

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

Methods Not Allowed

-none-

INVOKES_TOOL

Participants

Participant	Legal Values	Many
owner	TOOL	false
member	BINARY_TOOL TEXT_TOOL	true

MESSAGE—Message

Defines elements to represent operations that can be applied to repository elements. Each MESSAGE element defines the calling sequence for methods that respond to that message.

Summary

Symbolic Name	
C binding	MCS_elm_message
OpenVMS binding	MCS$r_elm_message
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 MESSAGE
Tag	NAD$K_ENT_MESSAGE

Defined Properties

argSpec

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)

MESSAGE

hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

new (VERSION)
promote (VERSION)
replace (VERSION)
reserve (VERSION)

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
free (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
purge (VERSION)
rename (NAMED_ELEMENT)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
verify (ELEMENT)

MESSAGE

Methods Not Allowed

unreserve

METHOD—Method

Defines elements to represent the procedures that implement messages for specific element types. They are invoked to perform actions in the repository as the result of sending messages to repository elements.

Summary

Symbolic Name	
C binding	MCS_elm_method
OpenVMS binding	MCS$r_elm_method
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 TOOL 4 METHOD
Tag	NAD$K_ENT_METHOD

Defined Properties

application
funcType—*Required*
implementsMessage
keepHist
methodType
postamble
preamble

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)

dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
invocationString (TOOL)—*Conditionally required*
invokes (TOOL)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

new (VERSION)
promote (VERSION)
replace (VERSION)
reserve (VERSION)

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
free (VERSION)
freeze (VERSION)
getProp (ELEMENT)

METHOD

merge (VERSION)
purge (VERSION)
rename (NAMED_ELEMENT)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
verify (ELEMENT)

Methods Not Allowed

unreserve

METHOD_INVOCATION—Method Invocation

Defines elements that represent the invocation of methods to build derived objects from source objects. These elements record dependencies between inputs, outputs, and processors.

Summary

Symbolic Name	
C binding	MCS_elm_method_invocation
OpenVMS binding	MCS$r_elm_method_invocation
Hierarchy Level	0 ELEMENT 1 METHOD_INVOCATION
Tag	NAD$K_ENT_METHOD_INVOC

Defined Properties

CPUTime
createdDate
derivedFrom
derives—*Required*
elapsedTime
invocationStatus
invocationString
logFile
optionsString
OSVersion
scalingFactor
toolVersion

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (ELEMENT)

METHOD_INVOCATION

getProp (ELEMENT)
new (ELEMENT)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

MSGARG—Message Argument

Defines elements that represent the characteristics of arguments that are passed with messages. A single MSGARG element may be shared by several messages.

Summary

Symbolic Name

C binding	MCS_elm_msgarg
OpenVMS binding	MCS$r_elm_msgarg
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 MSGARG
Tag	NAD$K_ENT_MSGARG

Defined Properties

dataType

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)

MSGARG

hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
free (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
new (VERSION)
promote (VERSION)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (VERSION)
reserve (VERSION)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
unreserve (VERSION)

verify (ELEMENT)

Methods Not Allowed

-none-

NAMED_ELEMENT—Named Element

Defines elements whose *individual instances* have unique names. All element types whose instances require names are subtypes of NAMED_ELEMENT.

Summary

Symbolic Name	
C binding	MCS_elm_named_element
OpenVMS binding	MCS$r_elm_named_element
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT
Tag	NAD$K_ENT_NAMED_ELE

Defined Properties

access
alternateNames
createdDate
description
history
name
owner
processingName
simpleName

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)

Defined Methods

rename

Refined Methods

new (ELEMENT)

Inherited Methods

free (ELEMENT)
getProp (ELEMENT)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

OPENED_BY—File Opened By

Defines relationships that associate a CONTEXT element with files (BINARY elements and subtypes) it has opened. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name	
C binding	MCS_elm_opened_by
OpenVMS binding	MCS$r_elm_opened_by
Hierarchy Level	0 ELEMENT 1 RELATION 2 OPENED_BY
Tag	NAD$K_REL_OPENED_BY

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	CONTEXT	true
member	BINARY	true

PARTITION—Partition

Defines elements to represent named repository divisions that contain VERSION elements. Each VERSION element exists in only one partition. Partitions are arranged in a hierarchy; a repository may contain one or more partition hierarchies.

Summary

Symbolic Name	
C binding	MCS_elm_partition
OpenVMS binding	MCS$r_elm_partition
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 PARTITION
Tag	NAD$K_ENT_PARTITION

Defined Properties

allChildPartitions
allParentPartitions
autopurge
childPartitions
instances
parentPartition
partitionDir
related

Inherited Properties

access (NAMED_ELEMENT)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
createdDate (NAMED_ELEMENT)
description (NAMED_ELEMENT)
elementType (ELEMENT)
history (NAMED_ELEMENT)
name (NAMED_ELEMENT)—*Required*
owner (NAMED_ELEMENT)
processingName (NAMED_ELEMENT)
simpleName (NAMED_ELEMENT)

Defined Methods

-none-

Refined Methods

free (ELEMENT)
new (NAMED_ELEMENT)
setProp (ELEMENT)

Inherited Methods

getProp (ELEMENT)
rename (NAMED_ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

PERSISTENT_PROCESS—Persistent Process

Defines elements that record the state of interactive sessions. PERSISTENT_PROCESS elements contain information about current defaults and settings both within the repository and in the operating system.

Summary

Symbolic Name	
C binding	MCS_elm_persistent_process
OpenVMS binding	MCS$r_elm_persistent_process
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 PERSISTENT_PROCESS
Tag	NAD$K_ENT_PERS_PROC

Defined Properties

aliases
currCollection
currContext
defaultAccess
symbols

Inherited Properties

access (NAMED_ELEMENT)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
createdDate (NAMED_ELEMENT)
description (NAMED_ELEMENT)
elementType (ELEMENT)
history (NAMED_ELEMENT)
name (NAMED_ELEMENT)—*Required*
owner (NAMED_ELEMENT)
processingName (NAMED_ELEMENT)
simpleName (NAMED_ELEMENT)

Defined Methods

close
open

PERSISTENT_PROCESS

Refined Methods

free (ELEMENT)
setProp (ELEMENT)

Inherited Methods

getProp (ELEMENT)
new (NAMED_ELEMENT)
rename (NAMED_ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

PROPERTY_TYPE—Property

Defines elements that store characteristics of properties, including name, access, and data type.

Summary

Symbolic Name

C binding	MCS_elm_property_type
OpenVMS binding	MCS$r_elm_property_type

Hierarchy Level

0 ELEMENT
  1 NAMED_ELEMENT
    2 VERSION
      3 TYPE
        4 PROPERTY_TYPE

Tag: NAD$K_ENT_ATT_TYPE

Defined Properties

accessType
dataType
scale

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
allSubTypes (TYPE)
allSuperTypes (TYPE)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)

description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
subTypes (TYPE)
superTypes (TYPE)
tag (TYPE)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

new (VERSION)

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
promote (TYPE)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (TYPE)
reserve (TYPE)

PROPERTY_TYPE

setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
verify (ELEMENT)

Methods Not Allowed

-none-

RELATION—Relation

Defines relationships, which associate other elements. RELATION defines the most general type of relationship.

Summary

Symbolic Name	
C binding	MCS_elm_relation
OpenVMS binding	MCS$r_elm_relation
Hierarchy Level	0 ELEMENT 1 RELATION
Tag	NAD$K_REL_REL

Defined Properties

relMember—*Required*
relOwner—*Required*

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)

Defined Methods

-none-

Refined Methods

free (ELEMENT)
new (ELEMENT)

Inherited Methods

getProp (ELEMENT)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

RELATION

Participants

Participant	Legal Values	Many
owner	ELEMENT	false
member	ELEMENT	false

RELATION_MEMBER—Relation Member

Defines relationships that connect RELATION_TYPE elements to ELEMENT_TYPE elements. A RELATION_MEMBER relationship indicates that elements of the element type may be members of relationships of the relation type. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name	
C binding	MCS_elm_relation_member
OpenVMS binding	MCS$r_elm_relation_member
Hierarchy Level	0 ELEMENT 1 RELATION 2 DEPENDS_ON 3 RELATION_MEMBER
Tag	NAD$K_REL_REL_MEMBER

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (DEPENDS_ON)
verify (ELEMENT)

RELATION_MEMBER

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	RELATION_TYPE	true
member	ELEMENT_TYPE	true

RELATION_TYPE—Relation Type

Defines elements that store the characteristics of relation types. Each relation type is an instance of RELATION_TYPE.

Summary

Symbolic Name	
C binding	MCS_elm_relation_type
OpenVMS binding	MCS$r_elm_relation_type
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 TYPE 4 ELEMENT_TYPE 5 RELATION_TYPE
Tag	NAD$K_ENT_RELATION_TYPE

Defined Properties

definedLegalMembers
definedLegalOwners
legalMembers—*Required*
legalOwners—*Required*

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
allInstances (ELEMENT_TYPE)
allSubTypes (TYPE)
allSuperTypes (TYPE)
alternateNames (NAMED_ELEMENT)
associatedValidations (ELEMENT_TYPE)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
compPropDef (ELEMENT_TYPE)
controlled (VERSION)

RELATION_TYPE

createdDate (NAMED_ELEMENT)
definedMethods (ELEMENT_TYPE)
definedPropDefs (ELEMENT_TYPE)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
instances (ELEMENT_TYPE)
lastVersion (VERSION)
methods (ELEMENT_TYPE)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
ownsRelation (ELEMENT_TYPE)
parentInContext (VERSION)
pattern (ELEMENT_TYPE)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
propDef (ELEMENT_TYPE)
relationMember (ELEMENT_TYPE)
relPropDef (ELEMENT_TYPE)
rootBranchInstances (ELEMENT_TYPE)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
subTypes (TYPE)
superTypes (TYPE)—*Required*
tag (TYPE)
versionable (ELEMENT_TYPE)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

new (ELEMENT_TYPE)

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
promote (TYPE)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (TYPE)
reserve (TYPE)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
verify (ELEMENT)

Methods Not Allowed

-none-

RESERVED_BY—Version Reserved By

Defines relationships that associate a reserved VERSION element with the CONTEXT element that owns the reservation. See Section A.1 for information about the properties that traverse these relationships.

Summary

Symbolic Name

C binding	MCS_elm_reserved_by
OpenVMS binding	MCS$r_elm_reserved_by
Hierarchy Level	0 ELEMENT 1 RELATION 2 RESERVED_BY
Tag	NAD$K_REL_RESERVATION

Defined Properties

-none-

Inherited Properties

allElementTypes (ELEMENT)
elementType (ELEMENT)
relMember—*Required* (RELATION)
relOwner—*Required* (RELATION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

free (RELATION)
getProp (ELEMENT)
new (RELATION)
setProp (ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

Participants

Participant	Legal Values	Many
owner	VERSION	true
member	CONTEXT	false

TEXT—Text File

Defines elements that represent text files. You can create subtypes of TEXT to represent specialized text files such as language source files, command scripts, and so on.

Summary

Symbolic Name
- C binding: MCS_elm_text
- OpenVMS binding: MCS$r_elm_text

Hierarchy Level:
0 ELEMENT
1 NAMED_ELEMENT
2 VERSION
3 AGGREGATE
4 BINARY
5 TEXT

Tag: NAD$K_ENT_TEXT

Defined Properties

-none-

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
deltaFile (BINARY)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)

elementType (ELEMENT)
filePath (BINARY)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
importedFrom (BINARY)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
openedBy (BINARY)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
referenceCount (BINARY)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
storedIn—*Conditionally required* (BINARY)
storeType—*Required* (BINARY)
versionNum (VERSION)

Defined Methods

differences

Refined Methods

edit (VERSION)
merge (VERSION)

Inherited Methods

attach (VERSION)
build (VERSION)
close (BINARY)
control (VERSION)
detach (BINARY)
export (BINARY)
free (BINARY)
freeze (VERSION)
getProp (ELEMENT)
import (BINARY)

TEXT

new (BINARY)
open (BINARY)
promote (BINARY)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (BINARY)
reserve (BINARY)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
unreserve (BINARY)
verify (BINARY)

Methods Not Allowed

-none-

TEXT_TOOL—Text Tool

Defines elements to represent scripts (internal or external) that implement methods.

Summary

Symbolic Name

C binding	MCS_elm_text_tool
OpenVMS binding	MCS$r_elm_text_tool

Hierarchy Level

```
0 ELEMENT
  1 NAMED_ELEMENT
    2 VERSION
      3 AGGREGATE
        4 BINARY
          5 TEXT
            6 TEXT_TOOL
```

Tag: NAD$K_ENT_TEXT_TOOL

Defined Properties

-none-

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
deltaFile (BINARY)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)

TEXT_TOOL

elementType (ELEMENT)
filePath (BINARY)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
importedFrom (BINARY)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
openedBy (BINARY)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
referenceCount (BINARY)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
storedIn—*Conditionally required* (BINARY)
storeType—*Required* (BINARY)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

attach (VERSION)
build (VERSION)
close (BINARY)
control (VERSION)
detach (BINARY)
differences (TEXT)
edit (TEXT)
export (BINARY)
free (BINARY)
freeze (VERSION)
getProp (ELEMENT)

import (BINARY)
merge (TEXT)
new (BINARY)
open (BINARY)
promote (BINARY)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (BINARY)
reserve (BINARY)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
unreserve (BINARY)
verify (BINARY)

Methods Not Allowed

-none-

TOOL—Tool

Defines elements to represent executable programs that can be invoked by a METHOD element.

Summary

Symbolic Name	
C binding	MCS_elm_tool
OpenVMS binding	MCS$r_elm_tool
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 TOOL
Tag	NAD$K_ENT_TOOL

Defined Properties

invocationString
invokes

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)

hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
free (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
new (VERSION)
promote (VERSION)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (VERSION)
reserve (VERSION)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
unreserve (VERSION)

TOOL

verify (ELEMENT)

Methods Not Allowed

-none-

TYPE—Type Definition

Defines elements that store characteristics of the types in the repository. Subtypes of TYPE define element types, data types, relation types, and properties.

Summary

Symbolic Name	
C binding	MCS_elm_type
OpenVMS binding	MCS$r_elm_type
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 TYPE
Tag	NAD$K_ENT_TYPE

Defined Properties

allSubTypes
allSuperTypes
subTypes
superTypes
tag

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)
derives (VERSION)

TYPE

description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
hasParents (VERSION)
history (NAMED_ELEMENT)
inPartition (VERSION)
lastVersion (VERSION)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

promote (VERSION)
replace (VERSION)
reserve (VERSION)

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
freeze (VERSION)
getProp (ELEMENT)
merge (VERSION)
new (VERSION)
purge (VERSION)
rename (NAMED_ELEMENT)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)

verify (ELEMENT)

Methods Not Allowed

free
unreserve

VALIDATION—Validation

Defines elements that represent user-supplied routines invoked by Oracle CDD/Repository to perform checks on operations.

Summary

Symbolic Name	
C binding	MCS_elm_validation
OpenVMS binding	MCS$r_elm_validation
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION 3 TOOL 4 METHOD 5 VALIDATION
Tag	NAD$K_ENT_VALIDATION

Defined Properties

validationAction
validationApply
validationQuery
validationWhen

Inherited Properties

access (NAMED_ELEMENT)
allDependencies (VERSION)
allDependents (VERSION)
allDerivedFrom (VERSION)
allDerives (VERSION)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
application (METHOD)
attachmentInContext (VERSION)
availVersion (VERSION)
branchName (VERSION)
controlled (VERSION)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
dependents (VERSION)
derivedFrom (VERSION)

derives (VERSION)
description (NAMED_ELEMENT)
elementType (ELEMENT)
firstVersion (VERSION)
freezeTime (VERSION)
funcType—*Required* (METHOD)
hasParents (VERSION)
history (NAMED_ELEMENT)
implementsMessage (METHOD)
inPartition (VERSION)
invocationString (TOOL)—*Conditionally required*
invokes (TOOL)
keepHist (METHOD)
lastVersion (VERSION)
methodType (METHOD)
name (NAMED_ELEMENT)—*Required*
nextVersions (VERSION)
owner (NAMED_ELEMENT)
parentInContext (VERSION)
postamble (METHOD)
preamble (METHOD)
prevVersions (VERSION)
processingName (NAMED_ELEMENT)
rootBranchName (VERSION)
rootVersion (VERSION)
simpleName (NAMED_ELEMENT)
status (VERSION)
versionNum (VERSION)

Defined Methods

-none-

Refined Methods

-none-

Inherited Methods

attach (VERSION)
build (VERSION)
control (VERSION)
detach (VERSION)
edit (VERSION)
free (VERSION)
freeze (VERSION)

VALIDATION

getProp (ELEMENT)
merge (VERSION)
new (METHOD)
promote (METHOD)
purge (VERSION)
rename (NAMED_ELEMENT)
replace (METHOD)
reserve (METHOD)
setProp (VERSION)
translate (VERSION)
unfreeze (VERSION)
verify (ELEMENT)

Methods Not Allowed

-none-

VERSION—Version

Defines elements possessing the properties and methods that implement the Oracle CDD/Repository versioning model. Subtypes of VERSION specialize and extend the model as appropriate for the objects they represent; but VERSION handles the basic management of relationships in the repository among a set of versions.

Summary

Symbolic Name	
C binding	MCS_elm_version
OpenVMS binding	MCS$r_elm_version
Hierarchy Level	0 ELEMENT 1 NAMED_ELEMENT 2 VERSION
Tag	NAD$K_ENT_VERSION

Defined Properties

allDependencies
allDependents
allDerivedFrom
allDerives
attachmentInContext
availVersion
branchName
controlled
dependencies
dependents
derivedFrom
derives
firstVersion
freezeTime
hasParents
inPartition
lastVersion
nextVersions
parentInContext
prevVersions
rootBranchName
rootVersion

VERSION

status
versionNum

Inherited Properties

access (NAMED_ELEMENT)
allElementTypes (ELEMENT)
alternateNames (NAMED_ELEMENT)
createdDate (NAMED_ELEMENT)
description (NAMED_ELEMENT)
elementType (ELEMENT)
history (NAMED_ELEMENT)
name (NAMED_ELEMENT)
owner (NAMED_ELEMENT)
processingName (NAMED_ELEMENT)
simpleName (NAMED_ELEMENT)

Defined Methods

attach
build
control
detach
edit
freeze
merge
promote
purge
replace
reserve
translate
unfreeze
unreserve

Refined Methods

free (ELEMENT)
new (NAMED_ELEMENT)
setProp (ELEMENT)

Inherited Methods

getProp (ELEMENT)
rename (NAMED_ELEMENT)
verify (ELEMENT)

Methods Not Allowed

-none-

2
Message Descriptions

This chapter contains descriptions of the messages provided with Oracle CDD/Repository, arranged alphabetically. Each message description contains the following sections:

Title—Includes the generic name of the message, a short phrase that describes the purpose, and a paragraph briefly describing effects and intended use. This description is general. To find out how each element type responds to the message, you must read the Methods section.

Symbolic Name—Gives the symbolic name of the message. Use the symbol when a routine call requires you to identify the message name.

Arguments—Defines message arguments. You supply arguments with messages in the form of an argument list. Arguments fall into three categories:

- Required—You must supply these arguments. Each method checks the argument list for required arguments and fails if they are not present. These arguments are documented in the Required Arguments section.
- Optional and used—You do not need to supply these arguments. If these arguments are present, the method will make use of them. If the arguments are absent, the method will supply default values (documented in the Methods section). These arguments are documented in the Optional Arguments section.

  Any time you send any message, you can include the optional *comment* argument. The value of this argument is a list of strings containing a comment. If the first method to respond to the message is set to create a history record for the operation (that is, the METHOD element's **keepHist** property is TRUE), the method dispatcher creates an EVENT element and associates it with the element affected or created by the operation. The EVENT element's **historyComment** property contains the comment that you supply with this argument.

- Optional and ignored—These arguments have no meaning to the methods supplied by Oracle CDD/Repository and are ignored. Application integrators defining subtypes and method refinements can use optional arguments.

  Any arguments on the argument list that do not fall into the Required or Optional and used categories are ignored.

Each message description contains lists of the required and optional arguments, if they exist. Each argument description consists of the following:

- The symbolic name of the argument as you specify it in the argument list.
- The argument's data type.
- The use of the argument by Oracle CDD/Repository. "Input" means that the argument is used as input to the method; "output" means that the method returns a value in the argument.

Method Summary—This section illustrates a subset of the Oracle CDD/Repository element type hierarchy, showing how each type responds to the message. Next to each type, one of five legends appears, as follows (legends appearing in bold type indicate that a method description appears in the Methods section):

- **Defines**—The type defines a method for the message. No supertype of this type recognizes the message.
- **Refines**—The type refines the method that it inherits from a supertype. By looking upwards in the type hierarchy, you can tell which type defines (or refines) the method that this type refines.
- **Disallows**—The type disallows the message.
- *Inherits*—The type inherits a supertype's method without refinement.
- *Does not recognize*—The type does not recognize the message. This implies that no supertype recognizes the message either.

If an element type does not recognize a message and none of its subtypes do either, the type and subtypes are not listed in this section. Therefore, you can assume that any element type not appearing in the listing does not recognize the message.

Methods—This section lists each element type noted in the Method Summary as defining, refining, or disallowing the message, a description of the method's processing steps, and a list of the error codes it can return. The processing descriptions use the following notational conventions:

- The word "*element*" in italics refers to the element to which the message was sent; in other words, the first argument to MCS_dispatch_op or the second argument to MCS_dispatch_superOp.
- The word "verify" indicates that the method performs an error check. If the error check fails, the method terminates and returns an error status. (The method may perform other error checks not listed in the description. For example, all methods verify that the sender has appropriate access to the elements they manipulate.)
- The word "Superop," listed as a separate step, indicates that the method calls the MCS_dispatch_superOp routine on *element* at this point, allowing the supertype's method to perform its processing. Read the description of the supertype's method to find out what takes place.
- If the method sends a message, that message is named and highlighted in the text. For example: "Send the **new** message to *element*'s type." When a method sends a message, find and read the description of the resulting method to find out what takes place.

The description of the method's processing is followed by a list of the errors that the method itself can generate. The list does not include the errors that supertype or other methods invoked by the described method can generate. In addition, each method performs checks for appropriate access on the elements it reads or modifies. These checks can result in errors that are the same for all methods and that are not described in this section.

In addition to the errors shown with each method, methods may return the following codes:

BADMALLOC	An error occurred while allocating virtual memory.
MISSING_ARGUMENT	For messages that have required arguments, one or more of the required arguments was omitted.
SUCCESS	Normal successful completion. All methods return this code if they execute normally, and do not put anything on the error stack.

Depending on the bindings you are using, you should precede the code names with either MCS_ or MCS$_ to form the symbolic name of the code: for example, MCS_SUCCESS, MCS$_SUCCESS.

In general, the errors listed in this section are those that the method will place on top of the error stack. In many cases, the listed error will be the return value of MCS_dispatch_op and will appear on top of the error stack following a failed call to MCS_dispatch_op. In other cases, MCS_dispatch_op will place its own error status on top of the error stack and return that value as its completion status. Examine the contents of the error stack to find out exactly what caused the error.

attach—Attach to Composite

Makes the VERSION element to which it is sent a member of a composite. The composite must be reserved.

Symbolic Name

C Binding: MCS_message_attach
OpenVMS Binding: MCS$r_message_attach

Required Arguments

MCS_arg_collection_elmID (C binding)
MCS$r_arg_collection_elmID (OpenVMS binding)
data type: MCS_ELEMENTID
use: input
Element ID of the COMPOSITE element to which the VERSION element is to be attached.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
  1 NAMED_ELEMENT *does not recognize*
    2 VERSION **defines**
      3 AGGREGATE *inherits*
        4 BINARY *inherits*
          5 BINARY_TOOL *inherits*
          5 TEXT *inherits*
            6 TEXT_TOOL *inherits*
        4 COMPOSITE *inherits*
          5 COLLECTION **refines**
      3 MESSAGE *inherits*
      3 MSGARG *inherits*
      3 TOOL *inherits*
        4 METHOD *inherits*

5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*
5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

COLLECTION

1. Verify that attaching *element* will not create a cycle in the collection hierarchy.
2. Superop.

Errors

CYCLE	The requested attachment would have created a cycle in the collection graph.

VERSION

1. Verify that the composite element specified on the argument list is actually a composite.
2. Verify that the composite is reserved and is owned by the user issuing the message.
3. Verify that no other version of the component that includes *element* is already a member of the composite.
4. Add *element* to the composite by creating a COMPOSITE_PART relationship from the composite to *element*.

Errors

ALRINCOLL	Another version of *element* is already in the composite.
NOCONTEXT	No context has been opened.
NOTCOLL	Element type is not COMPOSITE or subtype.
NOTRESERVED	The composite is not reserved in this context.

build—Build Current Version

Creates a new version of the element to which it is sent if the element represents an object that is out of date.

Note

The **build** message performs no action in this release of Oracle CDD/Repository. It acts as a placeholder for future releases.

Symbolic Name

C Binding: MCS_message_build
OpenVMS Binding: MCS$r_message_build

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION **defines**
3 AGGREGATE *inherits*
4 BINARY *inherits*
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE *inherits*
3 MSGARG *inherits*
3 TOOL *inherits*

4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*
5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

VERSION

For this version of Oracle CDD/Repository, this is a null method.

close—Close Element

Terminates access to the contents of the element to which it is sent. For element types in which opening the element provides exclusive access, closing the element makes it available for others to use.

Symbolic Name

C Binding: MCS_message_close
OpenVMS Binding: MCS$r_message_close

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 CONTEXT **defines**
2 PERSISTENT_PROCESS **defines**
2 VERSION *does not recognize*
3 AGGREGATE **defines**
4 BINARY **refines**
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION **refines**

close

Methods

AGGREGATE

Performs no processing. It is refined by subtypes of AGGREGATE.

BINARY

1. Superop.
2. If *element* is reserved or stored externally, terminate the method successfully.
3. Verify that *element* has been opened by the current context.
4. Delete the file system link(s) that point to *element*'s file from collections owned by the current context.
5. If the file is no longer opened by anyone, delete it.

Errors

BINARYNOTOPEN	*element* is not open in the current context.
NOCONTEXT	There is no current context.

COLLECTION

1. Verify that *element* is the currently opened collection.
2. Superop.
3. If there is a current persistent process, set the value of its **currCollection** property to NULL.

Errors

COLLNOTOPEN	*element* is not open.
PP_NOT_OPEN	No persistent process is currently open.

CONTEXT

1. Verify that *element* is open.
2. Set the current context to NULL.

Errors

CONTEXT_NOT_OPEN	Either *element* is already closed, or is open but is not the user's current context.

PERSISTENT_PROCESS

1. Verify that *element* is open.
2. If *element* has an associated context, send the **close** message to it.
3. Close *element*.

Errors

PP_NOT_OPEN	*element* is not open.

control—Control Element

Places all versions of the element to which it is sent under configuration control by moving them into the current base partition. Versionable elements that are created when a context is active are automatically created as controlled elements. If no context is active, these elements are created as uncontrolled elements. However, COLLECTION and BINARY elements and their subtypes cannot be created as uncontrolled elements.

Symbolic Name

C Binding: MCS_message_control
OpenVMS Binding: MCS$r_message_control

Required Arguments

None.

Optional Arguments

MCS_arg_closure (C binding)
MCS$r_arg_closure (OpenVMS binding)
data type: MCS_SMALLINT
use: input
Indicates whether other elements related to the element should be controlled. Values are shown in the following table:

Value	Meaning
MCS_TO_NONE	Control only this element. This is the default.
MCS_TO_BOTTOM	Control this element and all uncontrolled elements owned, either directly or indirectly, through dependency relationships. These include direct and indirect children of a composite.

MCS_arg_closure_list (C binding)
MCS$r_arg_closure_list (OpenVMS binding)
data type: MCS_LIST
use: output
Receives the list of elements which were controlled.

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION **defines**
3 AGGREGATE *inherits*
4 BINARY *inherits*
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE *inherits*
3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*
5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

VERSION

1. If the *closure* argument was included and its value is MCS_TO_BOTTOM, send the **control** message with the same *closure* argument value to each of *element*'s children.
2. Verify that *element* is uncontrolled (the value of its **controlled** property is FALSE).
3. For each version of *element*'s component, verify that all elements to which the version owns dependency relationships lie between the current base partition and the root of the partition hierarchy. This also means that the element may not own a dependency relationship to either an unversioned or uncontrolled element as well.

4. Set **inPartition** to the current base partition for each version of *element*'s component.
5. Set the value of the **controlled** property to TRUE.
6. If the *closure_list* argument was specified, add the controlled version to the list.

Errors

BADCLOS	A *closure* value other than TO_NONE or TO_BOTTOM was specified.
ISCONTROLLED	*element* is already controlled.
NOBASEPART	The method cannot find the base partition.
NOCONTEXT	No context has been opened.
NOTVERSIONED	*element* is not a versionable object.
NOTVISIBLE	*element* is not visible in this context.

detach—Detach from Composite

Removes the VERSION element to which it is sent from a composite. The composite must be reserved.

Symbolic Name

C Binding: MCS_message_detach
OpenVMS Binding: MCS$r_message_detach

Required Arguments

MCS_arg_collection_elmID (C binding)
MCS$r_arg_collection_elmID (OpenVMS binding)
data type: MCS_ELEMENTID
use: input
Element ID of the COMPOSITE element from which the VERSION element is to be detached.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
  1 NAMED_ELEMENT *does not recognize*
    2 VERSION **defines**
      3 AGGREGATE *inherits*
        4 BINARY **refines**
          5 BINARY_TOOL *inherits*
          5 TEXT *inherits*
            6 TEXT_TOOL *inherits*
        4 COMPOSITE *inherits*
          5 COLLECTION **refines**
      3 MESSAGE *inherits*
      3 MSGARG *inherits*
      3 TOOL *inherits*
        4 METHOD *inherits*

5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*
5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

BINARY

1. Verify that *element* is not open or reserved.
2. Superop.
3. If *element* has no more parents in context, close it.

Errors

ISRESERVED *element* represents a file that is still reserved.

COLLECTION

1. Superop.
2. If opened files now have no parent in context, close them.
3. If the current collection is no longer under the top collection, set the current collection to NULL.

VERSION

1. Verify that the specified composite is reserved.
2. Verify that the composite is owned by the user issuing the message.
3. If *element* is reserved, verify that it has more than one parent-in-context. (This ensures that *element* will not become an "orphan" when it is detached.)
4. Verify that *element* is a member of the composite.
5. Remove *element* from the composite.

Errors

ISRESERVED	*element* is reserved and cannot be detached from the composite.
NOTATTCH	*element* is not attached to the specified composite.
NOTCOLL	Element type is not COMPOSITE or subtype.
NOTRESERVED	The composite is not reserved in this context.
NOTUNDERTOP	The composite is not in the composite hierarchy identified by the current context.

differences—Differences

Performs a Differences operation on two files represented by TEXT elements (or subtypes), and places the difference records in a specified file in the native file system. If no differences are found, no file is created.

Symbolic Name

C Binding: MCS_message_differences
OpenVMS Binding: MCS$r_message_differences

Required Arguments

MCS_arg_fname (C binding)
MCS$r_arg_fname (OpenVMS binding)
data type: MCS_STRING
use: input
Name of the file in the native file system in which to place difference records. If there are no difference records, this file will not be created.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

MCS_arg_diff_elmID (C binding)
MCS$r_arg_diff_elmID (OpenVMS binding)
data type: MCS_ELEMENTID
use: input
Element ID of a TEXT (or subtype) element representing a file to be used as the second file in the Differences operation. If this argument is omitted, *element*'s predecessor is used.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION *does not recognize*
3 AGGREGATE *does not recognize*
4 BINARY *does not recognize*
5 TEXT **defines**
6 TEXT_TOOL *inherits*

Methods

TEXT

1. Verify that the *fname* argument was specified in the dispatch list.
2. If the *diff_elmID* argument was not specified, use the element ID for *element*'s predecessor.
3. Call the delta mechanism to compare *element* (as the first file) and either *diff_elmID* or *element*'s predecessor (the second file), specifying *fname* as the output file for difference records.

Errors

MISSING_ARGUMENT — A required argument was omitted, or the *diff_elmID* argument was omitted and *element* had no predecessor.

SUCCESS is returned if the operation completes successfully. The error stack contains one of the following entries to indicate the results:

DELTA$_DIFFERENT — Differences were found.

DELTA$_IDENTICAL — No differences were found.

edit—Edit

Makes the object represented by the element to which it is sent available to the user for editing. The message generally invokes an editing tool and loads the object.

Symbolic Name

C Binding: MCS_message_edit
OpenVMS Binding: MCS$r_message_edit

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION **defines**
3 AGGREGATE *inherits*
4 BINARY *inherits*
5 BINARY_TOOL *inherits*
5 TEXT **refines**
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE *inherits*
3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*

5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

TEXT

1. Invoke the default text editor. (Only operates if additional support for external program methods has been installed; fails otherwise.)

VERSION

No action. (This method is a placeholder for subtypes of VERSION.)

export—Export to File System

Copies to the native file system the contents of the object represented by the element to which it is sent.

Symbolic Name

C Binding: MCS_message_export
OpenVMS Binding: MCS$r_message_export

Required Arguments

MCS_arg_fname (C binding)
MCS$r_arg_fname (OpenVMS binding)
data type: MCS_STRING
use: input
Name of the file in the native file system.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION *does not recognize*
3 AGGREGATE **defines**
4 BINARY **refines**
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION **refines**

Methods

AGGREGATE

Performs no processing. This method is refined by subtypes of AGGREGATE.

BINARY

1. Superop.
2. Verify that *element* is stored internally.
3. Create a clear copy of the file represented by *element* with the name specified by the **export** message.

Errors

INVBRSYN	Invalid branch syntax.
INVSTORETYPE	*element* does not have a store type of INTERNAL.

COLLECTION

1. Superop.
2. Verify that the directory specified in the message argument list exists.
3. For each of *element*'s BINARY (and subtype) children:
   a. Construct a file specification from the directory specified in the message and the value of the **simpleName** property of the child.
   b. Send the **export** message to the child.

Errors

BADSTAT	An error occurred while trying to retrieve file attributes.
FILENOTDIR	The message specified a file instead of a directory or subdirectory.

free—Delete Element

Deletes the element to which it is sent.

Symbolic Name

C Binding: MCS_message_free
OpenVMS Binding: MCS$r_message_free

Required Arguments

None.

Optional Arguments

MCS_arg_closure (C binding)
MCS$r_arg_closure (OpenVMS binding)
data type: MCS_SMALLINT
use: input
Indicates how much of the collection containing the element should be freed. Values are shown in the following table:

Value	Meaning
MCS_TO_NONE	Free only this element. This is the default.
MCS_TO_BOTTOM	Free this element and all direct and indirect children in the collection hierarchy.

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT **defines**
  1 EVENT *inherits*
  1 METHOD_INVOCATION *inherits*
    2 ACAS_METHOD_INVOC *inherits*
    2 ATIS_METHOD_INVOC *inherits*
  1 NAMED_ELEMENT *inherits*
    2 CONTEXT **refines**

2 DATABASE **disallows**
2 DIRECTORY **refines**
2 PARTITION **refines**
2 PERSISTENT_PROCESS **refines**
2 VERSION **refines**
3 AGGREGATE *inherits*
4 BINARY **refines**
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION **refines**
3 MESSAGE *inherits*
3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE **disallows**
4 ELEMENT_TYPE **disallows**
1 RELATION **refines**
2 DEPENDS_ON *inherits*
3 COMPOSITE_PART *inherits*
3 HAS_DEFAULT_METHOD *inherits*
3 HAS_MESSAGE *inherits*
3 HAS_MSGARG *inherits*
3 HAS_PROPERTY *inherits*
4 HAS_COMPUTED_PROPERTY *inherits*
4 HAS_RELATION_PROPERTY *inherits*
3 HAS_RELATION *inherits*
3 HAS_SUPERTYPE *inherits*
3 IMPLEMENTS_METHOD *inherits*
3 IMPLEMENTS_RELATION *inherits*
3 INVOKES_TOOL *inherits*
3 RELATION_MEMBER *inherits*
2 HAS_CONTEXT *inherits*
2 HAS_CURR_COLLECTION *inherits*
2 HAS_DATATYPE *inherits*
2 HAS_PARENT *inherits*
2 HAS_POSTAMBLE *inherits*
2 HAS_PREAMBLE *inherits*
2 HAS_RELATED_PARTITION *inherits*
2 HAS_TOP_COLLECTION *inherits*

2 OPENED_BY *inherits*
2 RESERVED_BY *inherits*

Methods

BINARY

1. If *element* is stored internally, delete the file version represented by *element*.
2. Superop.

COLLECTION

1. If *element* is the top collection, set the current collection to NULL.
2. Superop.

CONTEXT

1. Verify that *element* is closed.
2. Verify that *element* has no reserved or open files.
3. Recursively delete all subdirectories of *element*'s context directory.
4. Delete *element*'s context directory.
5. Superop.

Errors

HASCHECKOUTS	*element* has reserved elements.
NOTCLOSED	*element* is open.

DATABASE

Disallowed. A repository cannot be deleted while a transaction is open. Since sending any message requires that a transaction be open, it is impossible to send the **free** message to delete a repository. Use the MCS_DB_free routine instead.

DIRECTORY

1. Verify that the directory is empty (contains no names).

2. Delete the directory file in the file system.
3. Delete the DIRECTORY element.

Note that this method does not invoke the superop.

ELEMENT

1. If the *closure* argument was included and its value is MCS_TO_BOTTOM, send the **free** message with the same *closure* argument value to each of *element*'s children.
2. Verify that *element* is not a member of a relationship. (If *element* is a member of a COMPOSITE_PART relationship owned by an object freed as part of a closure in the previous step, this check is not performed.)
3. Free the storage occupied by *element*'s **history** property.
4. For each relationship owned by *element*: if *element* is the only owner, send the **free** message to the relationship; otherwise, remove *element* from the scan of owners.
5. Free *element*'s storage.

Errors

ERRDELFIRST	*element* is the first version on a line of descent and has successors on that line of descent.
NOTBOTTOM	A *closure* value other than TO_BOTTOM or TO_NONE was supplied.

ELEMENT_TYPE

Disallowed.

PARTITION

1. Verify that *element* has no child partitions.
2. Verify that *element* contains no versions.
3. Verify that no contexts have *element* as their base partition.
4. Superop.

Errors

PARTNOTEMPTY *element* indicates a partition that is not empty.

PERSISTENT_PROCESS

1. Verify that *element* is closed.
2. Superop.

Errors

PP_OPEN *element* indicates a persistent process that is open.

RELATION

1. Verify that none of *element*'s members are dependent, that is, have no names and no way of reaching them other than through *element*.
2. Superop.

TYPE

Disallowed.

VERSION

1. Verify that *element* is not reserved.
2. Verify that *element* has no branches descending from it and no branches merging into it.
3. Verify that *element* is not the first version on a branch with more than one version.
4. If *element* has a descendant, make it the direct descendant of *element*'s ancestor. (This cuts *element* out of the line of descent.)
5. Superop.

Errors

ISRESERVED *element* is currently reserved and cannot be deleted.

freeze—Freeze Version

Puts the VERSION element to which it is sent into the frozen state. A frozen version cannot be reserved.

Symbolic Name

C Binding: MCS_message_freeze
OpenVMS Binding: MCS$r_message_freeze

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION **defines**
3 AGGREGATE *inherits*
4 BINARY *inherits*
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE *inherits*
3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*

5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

VERSION

1. Verify that *element* is not reserved.
2. Set the value of **status** to MCS_STS_FROZEN.

Errors

CANTFREEZE	*element* cannot be frozen.
NOCONTEXT	There is no currently open context.
NOTVERSIONED	*element* is not a versioned element.

getProp—Get Property Values

Returns the type and value of one or more specified properties belonging to the element to which it is sent.

Symbolic Name

C Binding: MCS_message_getProp
OpenVMS Binding: MCS$r_message_getProp

Required Arguments

MCS_arg_arglist (C binding)
MCS$r_arg_arglist (OpenVMS binding)
data type: MCS_LIST
use: input-output
List of properties to be read. Each element of the argument list contains three items:

- Property name: The name of the property to be returned.
- Property value: Oracle CDD/Repository returns the value of the property in this item.
- Status: Oracle CDD/Repository fills in a code indicating that the property value was successfully retrieved (MCS_SUCCESS), that the property has never been set (MCS_MISSING), or that an error occurred.

Use the various MCS_arglist routines to access the elements of an argument list.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

getProp

Method Summary

0 ELEMENT **defines**
- 1 EVENT *inherits*
- 1 METHOD_INVOCATION *inherits*
  - 2 ACAS_METHOD_INVOC *inherits*
  - 2 ATIS_METHOD_INVOC *inherits*
- 1 NAMED_ELEMENT *inherits*
  - 2 CONTEXT *inherits*
  - 2 DATABASE *inherits*
  - 2 DIRECTORY *inherits*
  - 2 PARTITION *inherits*
  - 2 PERSISTENT_PROCESS *inherits*
  - 2 VERSION *inherits*
    - 3 AGGREGATE *inherits*
      - 4 BINARY *inherits*
        - 5 BINARY_TOOL *inherits*
        - 5 TEXT *inherits*
          - 6 TEXT_TOOL *inherits*
      - 4 COMPOSITE *inherits*
        - 5 COLLECTION *inherits*
    - 3 MESSAGE *inherits*
    - 3 MSGARG *inherits*
    - 3 TOOL *inherits*
      - 4 METHOD *inherits*
        - 5 VALIDATION *inherits*
    - 3 TYPE *inherits*
      - 4 DATA_TYPE *inherits*
      - 4 ELEMENT_TYPE *inherits*
        - 5 RELATION_TYPE *inherits*
      - 4 PROPERTY_TYPE *inherits*
- 1 RELATION *inherits*
  - 2 DEPENDS_ON *inherits*
    - 3 COMPOSITE_PART *inherits*
    - 3 HAS_DEFAULT_METHOD *inherits*
    - 3 HAS_MESSAGE *inherits*
    - 3 HAS_MSGARG *inherits*
    - 3 HAS_PROPERTY *inherits*
      - 4 HAS_COMPUTED_PROPERTY *inherits*
      - 4 HAS_RELATION_PROPERTY *inherits*
    - 3 HAS_RELATION *inherits*
    - 3 HAS_SUPERTYPE *inherits*
    - 3 IMPLEMENTS_METHOD *inherits*

3 IMPLEMENTS_RELATION *inherits*
3 INVOKES_TOOL *inherits*
3 RELATION_MEMBER *inherits*
2 HAS_CONTEXT *inherits*
2 HAS_CURR_COLLECTION *inherits*
2 HAS_DATATYPE *inherits*
2 HAS_PARENT *inherits*
2 HAS_POSTAMBLE *inherits*
2 HAS_PREAMBLE *inherits*
2 HAS_RELATED_PARTITION *inherits*
2 HAS_TOP_COLLECTION *inherits*
2 OPENED_BY *inherits*
2 RESERVED_BY *inherits*

Methods

ELEMENT

1. For each argspec in the arglist:
   a. Verify that *element* possesses the requested property.
   b. If the property is a computed property, invoke the method that computes it. Otherwise, obtain the value of the property and set the return value to it.
   c. If the value of the property was obtained successfully, set the *status* field of the argument to SUCCESS. If the property has no value, set the field to MISSING. Otherwise, set the field to a failure code.

Errors

ILLPRODEF	Illegal property definition.
MISSING	Requested property does not have a defined value.
NOTAPPLICABLE	*element* does not possess the requested property.
SOMEFAILED	One or more property values could not be retrieved.

import—Import from File System

Replaces the contents of the AGGREGATE element to which it is sent from the native file system. The element must be reserved.

Symbolic Name

C Binding: MCS_message_import
OpenVMS Binding: MCS$r_message_import

Required Arguments

MCS_arg_fname (C binding)
MCS$r_arg_fname (OpenVMS binding)
data type: MCS_STRING
use: input
Name of the file in the native file system.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION *does not recognize*
3 AGGREGATE **defines**
4 BINARY **refines**
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION **refines**

Methods

AGGREGATE

1. Verify that *element* is reserved.

Errors

NOTRESERVED *element* is not reserved.

BINARY

1. Superop.
2. Verify that *element* is stored internally.
3. Copy the file specified in the **import** message to the file specification contained in *element*'s **filePath** property.
4. Set *element*'s **importedFrom** property to the file specification supplied with the **import** message.

Errors

INVSTORETYPE *element*'s storage type is not INTERNAL.

COLLECTION

1. Superop.
2. Verify that the directory specified in the message argument list exists.
3. For each of *element*'s reserved BINARY (and subtype) children:
   a. Construct a file specification from the directory specified in the message and the value of the child's **simpleName** property.
   b. Send the **import** message to the child.

merge—Merge Versions

Merges one element into another, allowing two lines of descent to be merged into one. The VERSION element to which the message is sent is the version that is merged into, and is called the *source* of the merge. Since the merge operation changes the *source*, it must be reserved. The *merge* element specified in the argument list is the element that merges into the *source*.

Symbolic Name

C Binding: MCS_message_merge
OpenVMS Binding: MCS$r_message_merge

Required Arguments

MCS_arg_merge_elmID (C binding)
MCS$r_arg_merge_elmID (OpenVMS binding)
data type: MCS_ELEMENTID
use: input
VERSION element that will be merged with the source VERSION element.

Optional Arguments

MCS_arg_ancestor_elmID (C binding)
MCS$r_arg_ancestor_elmID (OpenVMS binding)[1]
data type: MCS_ELEMENTID
use: output
Element ID of the common ancestor of the *source* and *merge* composites.

MCS_arg_ancestor_list (C binding)
MCS$r_arg_ancestor_list (OpenVMS binding)[1]
data type: MCS_LIST
use: output
List of element IDs of children of the *ancestor* composite that match one for one with entries in the other lists. If there is no corresponding child, the element ID is null.

[1] This argument is used by COMPOSITE's refinement of **merge**. If omitted by the caller, it is added to the argument list by the refinement.

[1] This argument is used by COMPOSITE's refinement of **merge**. If omitted by the caller, it is added to the argument list by the refinement.

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

MCS_arg_merge_conflicts (C binding)
MCS$r_arg_merge_conflicts (OpenVMS binding)[1]
data type: MCS_LONGINT
use: output
Number of separate locations in the text where the changes conflicted and no merge was performed. These locations are flagged in the body of the text.

MCS_arg_merge_list (C binding)
MCS$r_arg_merge_list (OpenVMS binding)[2]
data type: MCS_LIST
use: output
List of element IDs of children of the *merge* composite that match one for one with entries in the other lists. If there is no corresponding child, the element ID is null.

MCS_arg_merge_successes (C binding)
MCS$r_arg_merge_successes (OpenVMS binding)[1]
data type: MCS_LONGINT
use: output
Number of separate locations in the text where changes were successfully merged.

MCS_arg_reason_list (C binding)
MCS$r_arg_reason_list (OpenVMS binding)[2]
data type: MCS_LIST
use: output
List of MCS_SMALLINT values that match one for one with entries in the other lists and that provide the reason for choosing each entry on the *result* list. Values are shown in the following table:

---

[1] This argument is used by TEXT's refinement of **merge**. If omitted by the caller, it is added to the argument list by the refinement.

[2] This argument is used by COMPOSITE's refinement of **merge**. If omitted by the caller, it is added to the argument list by the refinement.

Value	Meaning
MCS_MERGE_CONFLICT	The *source* and *merge* children are different from both the *ancestor* child and each other. Consequently, a conflict resulted and the corresponding *result* entry is null.
MCS_MERGE_MERGE_ CHANGED	The child of the *merge* composite is different from the corresponding child of the *ancestor* and *source* composites. Consequently, the *merge* child was selected for the *result* list.
MCS_MERGE_SOURCE_ CHANGED	The child of the *source* composite is different from the corresponding child of the *ancestor* and *merge* composites. Consequently, the *source* child was selected for the *result* list.
MCS_MERGE_UNCHANGED	The child is the same in the *ancestor*, *source*, and *merge* lists.

MCS_arg_result_list (C binding)
MCS$r_arg_result_list (OpenVMS binding)[1]
data type: MCS_LIST
use: output
List of element IDs that match one for one with entries in the other lists and that are the result of merging the *ancestor* list, the *merge* list, and the *source* list. If a conflict resulted from the **merge** operation, or if the result of the operation was such that there is no corresponding entry, the list contains a null element ID.

MCS_arg_source_list (C binding)
MCS$r_arg_source_list (OpenVMS binding)[1]
data type: MCS_LIST
use: output
List of element IDs of children of the *source* composite that match one for one with entries in the other lists. If there is no corresponding child, the element ID is null.

[1] This argument is used by COMPOSITE's refinement of **merge**. If omitted by the caller, it is added to the argument list by the refinement.
[1] This argument is used by COMPOSITE's refinement of **merge**. If omitted by the caller, it is added to the argument list by the refinement.

Method Summary

0 ELEMENT *does not recognize*
  1 NAMED_ELEMENT *does not recognize*
    2 VERSION **defines**
      3 AGGREGATE *inherits*
        4 BINARY *inherits*
          5 BINARY_TOOL *inherits*
          5 TEXT **refines**
            6 TEXT_TOOL *inherits*
        4 COMPOSITE **refines**
          5 COLLECTION *inherits*
      3 MESSAGE *inherits*
      3 MSGARG *inherits*
      3 TOOL *inherits*
        4 METHOD *inherits*
          5 VALIDATION *inherits*
      3 TYPE *inherits*
        4 DATA_TYPE *inherits*
        4 ELEMENT_TYPE *inherits*
          5 RELATION_TYPE *inherits*
        4 PROPERTY_TYPE *inherits*

Methods

COMPOSITE

1. Superop.
2. Determine the common ancestor version of the *source* composite (this is *element*) and the *merge* composite (supplied on the message argument list).
3. Determine the contents of the returned *result* and *reason* lists by comparing the corresponding children (those with the same root version names) of the *source*, *merge*, and *ancestor* composites, using the following rules:

   If *ancestor* child does not equal *source* child:

   If *ancestor* child does not equal *merge* child:

   If *source* child equals *merge* child, put *source* child in *result* list and SOURCE_CHANGED in *reason* list.

   Otherwise, put null in *result* list and CONFLICT in *reason* list.

   Otherwise, put *source* child in *result* list and SOURCE_CHANGED in *reason* list.

Otherwise:

If *ancestor* child does not equal *merge* child, put *merge* child in *result* list and MERGE_CHANGED in *reason* list.

Otherwise, put *ancestor* child in *result* list and UNCHANGED in *reason* list.

Note

- If there is no corresponding child in the *ancestor*, *source*, or *merge* composites, comparisons are made against the null elmID.
- Children in two composites are "equal" if they are the same element (have the same elmID).

Errors

NOTRESERVED *element* is not reserved.

TEXT

1. Verify that *element* is not stored externally.
2. Superop.
3. Invoke a text merge tool to merge the text files represented by the predecessor of *element* and the *merge* element, specifying that the output of the tool be placed in the file represented by *element*.

Errors

BADSTORETYPE *element* has a store type other than INTERNAL.

VERSION

The version merged into is *element*. The version that merges into *element* is specified on the argument list and is called *merge*.

1. Verify that *element* and *merge* have the same root version.
2. Verify that the component is controlled.
3. Verify that *element* is reserved.

4. Delete a MERGE_TO link to *element*, if one exists.
5. Create a MERGE_TO link from *merge* to *element*.

Errors

NOTRESERVED	*element* is not reserved.
NOTSAMEBR	The *source* and *merge* elements do not have the same root branch.

new—Create New Element

Creates a new instance of an element type and initializes its properties to values specified in the message, or (if not specified) to their default values.

Symbolic Name

C Binding: MCS_message_new
OpenVMS Binding: MCS$r_message_new

Required Arguments

MCS_arg_arglist (C binding)
MCS$r_arg_arglist (OpenVMS binding)
data type: MCS_LIST
use: input
List of properties to be set. Each element of the argument list contains three items:

- Property name—The name of the property to initialize.
- Property value—The value to which to initialize the property.
- Status—Oracle CDD/Repository returns a code indicating whether the property was successfully initialized.

Note

The *MCS_arg_arglist* argument is required only if the element being created has properties whose values must be specified with **new**. Refer to the individual element type descriptions for this information.

MCS_arg_new_inst_elmID (C binding)
MCS$r_arg_new_inst_elmID (OpenVMS binding)
data type: MCS_ELEMENTID
use: output
Element ID of the new element.

Optional Arguments

MCS_arg_collection_elmID (C binding)
MCS$r_arg_collection (OpenVMS binding)
data type: MCS_ELEMENTID
use: input
Element ID of a COMPOSITE to which to attach the new element. Used by VERSION's refinement to **new**.

MCS_arg_near_elmID (C binding)
MCS$r_arg_near_elmID (OpenVMS binding)
data type: MCS_ELEMENTID
use: input
Element ID of an element near which the new element should be stored.

Method Summary

0 ELEMENT **defines**
1 EVENT **refines**
1 METHOD_INVOCATION *inherits*
2 ACAS_METHOD_INVOC *inherits*
2 ATIS_METHOD_INVOC *inherits*
1 NAMED_ELEMENT **refines**
2 CONTEXT **refines**
2 DATABASE **disallows**
2 DIRECTORY **refines**
2 PARTITION **refines**
2 PERSISTENT_PROCESS *inherits*
2 VERSION **refines**
3 AGGREGATE *inherits*
4 BINARY **refines**
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE **refines**
3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD **refines**
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE **disallows**

4 ELEMENT_TYPE **refines**
5 RELATION_TYPE **refines**
4 PROPERTY_TYPE **refines**
1 RELATION **refines**
2 DEPENDS_ON *inherits*
3 COMPOSITE_PART *inherits*
3 HAS_DEFAULT_METHOD *inherits*
3 HAS_MESSAGE *inherits*
3 HAS_MSGARG *inherits*
3 HAS_PROPERTY *inherits*
4 HAS_COMPUTED_PROPERTY *inherits*
4 HAS_RELATION_PROPERTY *inherits*
3 HAS_RELATION *inherits*
3 HAS_SUPERTYPE *inherits*
3 IMPLEMENTS_METHOD *inherits*
3 IMPLEMENTS_RELATION *inherits*
3 INVOKES_TOOL *inherits*
3 RELATION_MEMBER *inherits*
2 HAS_CONTEXT *inherits*
2 HAS_CURR_COLLECTION *inherits*
2 HAS_DATATYPE *inherits*
2 HAS_PARENT *inherits*
2 HAS_POSTAMBLE *inherits*
2 HAS_PREAMBLE *inherits*
2 HAS_RELATED_PARTITION *inherits*
2 HAS_TOP_COLLECTION *inherits*
2 OPENED_BY *inherits*
2 RESERVED_BY *inherits*

Methods

BINARY

1. If the value for the **storeType** property supplied on the argument list is MCS_STORETYPE_EXTERNAL, verify that the argument list includes a value for the **storedIn** property.

   If the value for the **storeType** property is MCS_STORETYPE_INTERNAL, compute a name for the delta file that contains this file version and add it to the argument list as the value for the **deltaFile** property. If the argument list includes a value for the **importedFrom** property, the delta file is created from this value, which is a file specification.

2. Superop.

Errors

BADSTORETYPE	An illegal value for the **storeType** property was supplied.
ERRDELTALIB	An error occurred while creating a delta file subdirectory.
ERRMISSPROP	A property required when creating elements of this type was not specified. There are two possibilities: • Neither the **storedIn** property nor the **storeType** property was specified. • A value of MCS_STORETYPE_EXTERNAL was supplied for the **storeType** property and no value was supplied for the **storedIn** property.
ERRSETPROP	A value of MCS_STORETYPE_INTERNAL was supplied for the **storeType** property, and a value was supplied for the **storedIn** property. When creating internally stored BINARY elements, you must not supply a value for the **storedIn** property.
NOCONTEXT	No context is open.

CONTEXT

1. If the argument list does not include a value for the **contextDir** property, create a directory name based on the name specified for the new context, and add it to the argument list.
2. If the argument list does not include a value for the **defaultAttachment** property, supply a value of MCS_ATTACH_LATEST.
3. Superop.
4. Create the context directory.

Errors

BADDIRNAME	The value supplied as the value of the **contextDir** property could not be used to construct a valid directory name.
DIREXISTS	The value supplied as the value of the **contextDir** property specifies a directory that already exists.
ISRESERVED	The element ID supplied as the value of the **top** property identifies a reserved element.
NULLPROP	A null value was specified for a property that must have a non-null value.
REQATTMIS	No value was supplied for the **name** property.

DATABASE

Disallowed. Use MCS_DB_new instead.

DATA_TYPE

Disallowed. Oracle CDD/Repository does not allow user-defined data types. Therefore, the **new** message is disallowed on DATA_TYPE.

DIRECTORY

1. Verify that a value for the **name** property has been supplied.
2. Create the directory file in the file system.
3. Create the DIRECTORY element.

Note that this method does not invoke the superop.

ELEMENT

1. Verify that values have been supplied for all properties required by the type being instantiated.
2. Allocate storage for the new element.
3. Set the values of properties included in the argument list by sending the **setProp** message.

Errors

ERR_INST_NOT_TYPE	*element* is not of type TYPE or one of its subtypes.

ELEMENT_TYPE

1. If *element* is the ELEMENT_TYPE element named **ELEMENT_TYPE**, create the new element type as if it were a new version, but *do not superop*. Set up appropriate system properties. Generate a unique **tag** value if none was specified. Enter the element type name in the CDD$PROTOCOLS directory.

   Otherwise, invoke the **new** method for *element*.

Errors

DIR_PATH_IN_NAME	The name of the new element type included a directory path. Since metadata elements must occupy a specific directory, you should not include any directory information in the element's name.
ERR_INST_NOT_TYPE	*element* is not of type TYPE or one of its subtypes.
WRONG_DIC_CREATE	An attempt was made to create an element in one repository using an element type definition from a second repository.

EVENT

1. Add the current date and time to the argument list as the value for the **datetime** property.
2. Superop.

MESSAGE

1. Verify that the argument list includes a value for the **argSpec** property.
2. Superop.

Errors

DIR_PATH_IN_ NAME	The name of the new element type included a directory path. Since metadata elements must occupy a specific directory, you should not include any directory information in the element's name.

METHOD

1. If the argument list does not include a value for the **keepHist** property, add it to the argument list with a value of TRUE.
2. If the argument list does not include a value for the **invocationString** property, add it to the argument list with a null-string value.
3. If the value of the **funcType** property is MCS_METHOD_EXTERNAL_ CODE, MCS_METHOD_INTERNAL_CODE, or MCS_METHOD_ EXTERNAL_PROGRAM, verify that the value of the **invocationString** property is not a null string.
4. Superop.

Errors

DIR_PATH_IN_ NAME	The name of the new element type included a directory path. Since metadata elements must occupy a specific directory, you should not include any directory information in the element's name.

NAMED_ELEMENT

1. Verify that the specified value for the **name** property is not the name of an existing element.
2. Superop.

PARTITION

1. If the argument list does not include a value for the **partitionDir** property, supply a computed value.
2. Superop.
3. Create the partition directory.

Errors

BADDIRNAM	The value supplied for the **partitionDir** directory is an invalid directory name.
CYCLE	Creating the partition with the specified property values would create a cycle in the partition hierarchy.
DIREXISTS	The value supplied for the **partitionDir** directory specifies an existing directory.
REQATTMIS	No value was supplied for the **name** property.

PROPERTY_TYPE

1. Create the new property type as if it were a new version, but *do not superop*. Set up appropriate system properties. Generate a unique **tag** value if none was specified. Enter the property type name in the CDD$PROTOCOLS directory.

Errors

DIR_PATH_IN_ NAME	The name of the new element type included a directory path. Since metadata elements must occupy a specific directory, you should not include any directory information in the element's name.
DUPDATATYPES	A value was specified for both the **dataType** property and either the CDD$DATATYPE or CDD$SUBTYPE attribute.
WRONG_DIC_ CREATE	An attempt was made to create an element in one repository using an element type definition from a second repository.

RELATION

1. Verify that each of the members specified for the **relMember** property is of a type allowed by the **legalMembers** property of the relation's type.
2. Verify that each of the owners specified for the **relOwner** property is of a type allowed by the **legalOwners** property of the relation's type.
3. Superop.

RELATION_TYPE

1. If *element* is the ELEMENT_TYPE element named **RELATION_TYPE**, create the new relation type as if it were a new version, but *do not superop*. Set up appropriate system properties. Generate a unique **tag** value if none was specified. Enter the relation type name in the CDD$PROTOCOLS directory.

   Otherwise, create a new instance of an existing relation type.

Errors

DIR_PATH_IN_ NAME	The name of the new element type included a directory path. Since metadata elements must occupy a specific directory, you should not include any directory information in the element's name.
WRONG_DIC_ CREATE	An attempt was made to create an element in one repository using an element type definition from a second repository.

VERSION

1. Attempt to determine a parent composite to contain the new version. In the order in which they are checked, the parent is:
   a. A composite that was specified on the arglist.
   b. The current composite belonging to a persistent process that exists for this session.
   c. The composite identified by the **top** property of the current context.
2. If a parent has been identified in the previous steps, verify that it is reserved.
3. Superop.
4. If a parent has been identified:
   a. Add the new element to the parent's composite.
   b. Set the **inPartition** property on the new element to the current base partition.
5. Set the **status** property on the new element to MCS_STS_AVAIL.

Errors

COLLNOTCHILD	The composite to which to attach the new version is not in the hierarchy headed by the current top composite.
NOTCOLL	The element specified in the *collection_elmID* argument is not a subtype of COMPOSITE.

open—Open Element

Provides access to the contents of the element to which it is sent.

Symbolic Name

C Binding: MCS_message_open
OpenVMS Binding: MCS$r_message_open

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
  1 NAMED_ELEMENT *does not recognize*
    2 CONTEXT **defines**
    2 PERSISTENT_PROCESS **defines**
    2 VERSION *does not recognize*
      3 AGGREGATE **defines**
        4 BINARY **refines**
          5 BINARY_TOOL *inherits*
          5 TEXT *inherits*
            6 TEXT_TOOL *inherits*
        4 COMPOSITE *inherits*
          5 COLLECTION **refines**

Methods

AGGREGATE

Performs no processing. It is refined by subtypes of AGGREGATE.

BINARY

1. Superop.
2. If *element* is stored externally or is reserved, terminate the method.
3. If the current context already has opened *element*, terminate the method.
4. If *element* has not already been opened by another context, fetch the file it represents into the current partition directory.
5. Enter a symbolic file link to the file into the current context directory/directories.

Errors

BINARYALREADY-OPEN	*element* is already open.
ERROPENNOTINCTX	*element* is not in the current context.
ERROPENNOTINCTX	*element* is not in the current context.
NOCONTEXT	There is no current context.

COLLECTION

1. Superop.
2. Verify that no collection is already open in the current persistent process.
3. Set the value of the **currCollection** property for the current persistent process to *element*.

open

Errors

ANOTHERCOLLOPEN	Another collection is already the value of the **currCollection** property for this persistent process.
COLLALREADYOPEN	The specified collection is already the value of the **currCollection** property for this persistent process.
COLLNOTCHILD	The specified collection is not in the composite hierarchy headed by the current top collection.
PP_NOT_OPEN	There is no current persistent process.

CONTEXT

1. Verify that *element* is closed.
2. Set *element* as the current context.
3. Set the value of the **currContext** property on the current persistent process to *element*.

Errors

ERROPENCTX	An error occurred while trying to open *element*.
NOTCLOSED	*element* is already open.

PERSISTENT_PROCESS

1. Verify that there is no current persistent process.
2. If *element* has an associated context, send the **open** message to it.
3. Open *element*.

Errors

PP_ALREADY_OPEN	*element* is already open.
PP_NOCONTEXT	There is no current context and *element* has no associated context.

promote—Promote to Next Partition

Promotes the VERSION element to which it is sent to the next-higher partition in the partition hierarchy.

Symbolic Name

C Binding: MCS_message_promote
OpenVMS Binding: MCS$r_message_promote

Required Arguments

None.

Optional Arguments

MCS_arg_closure (C binding)
MCS$r_arg_closure (OpenVMS binding)
data type: MCS_SMALLINT
use: input
Indicates whether other elements related to the element should be promoted. Values are shown in the following table:

Value	Meaning
MCS_TO_NONE	Promote only this element. This is the default.
MCS_TO_BOTTOM	Promote this element and all elements that reside in the same partition and that it owns, either directly or indirectly, through dependency relationships. These include direct and indirect children of a composite.

MCS_arg_closure_list (C binding)
MCS$r_arg_closure_list (OpenVMS binding)
data type: MCS_LIST
use: output
Receives the list of elements that were promoted.

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

promote

Method Summary

0 ELEMENT *does not recognize*
  1 NAMED_ELEMENT *does not recognize*
    2 VERSION **defines**
      3 AGGREGATE *inherits*
        4 BINARY **refines**
          5 BINARY_TOOL *inherits*
          5 TEXT *inherits*
            6 TEXT_TOOL *inherits*
        4 COMPOSITE *inherits*
          5 COLLECTION **refines**
      3 MESSAGE **refines**
      3 MSGARG *inherits*
      3 TOOL *inherits*
        4 METHOD **refines**
          5 VALIDATION *inherits*
      3 TYPE **refines**
        4 DATA_TYPE *inherits*
        4 ELEMENT_TYPE *inherits*
          5 RELATION_TYPE *inherits*
        4 PROPERTY_TYPE *inherits*

Methods

BINARY

1. Superop.
2. If the file is open, make it accessible from the partition to which *element* has been promoted.

COLLECTION

1. If *element* is the CDD$METADATA collection and it is being promoted into CDD$METADATA_PARTITION, rebuild the dictionary metadata schema. Following this, the metadata elements contained in the collection become the current dictionary schema.
2. Superop.

MESSAGE

1. Verify that the current context's **top** property is set to the CDD$METADATA collection.
2. If the *closure* argument is not supplied or is supplied as MCS_TO_NONE, set it to MCS_TO_BOTTOM. In any case, make sure *closure* is set such that the entire metadata collection will be promoted.
3. Superop.

Note

The "metadata collection" is not a true collection in that it is not connected through COMPOSITE_PART subtype relationships. However, this method treats it as a collection.

Errors

BADARGUMENT	An invalid argument was specified.
BADCLOS	An inappropriate value was supplied for the *closure* argument.
METATOP	The CDD$METADATA collection is not at the top of the collection hierarchy.

METHOD

See the description of the **promote** method defined by MESSAGE, which is identical to this method.

TYPE

See the description of the **promote** method defined by MESSAGE, which is identical to this method.

VERSION

1. If the *closure* argument was included and its value is MCS_TO_BOTTOM, send the **promote** message with the same *closure* argument value to each of *element*'s children.

2. Verify that *element* is not reserved. (If the method was called recursively to promote all or part of a collection, and *element* is reserved, the method simply adds *element* to the *closure_list*, if specified, and exits.)
3. Verify that *element* is in a partition (other than the root partition) that lies in the path from the current context to the root partition.
4. Verify that *element* owns no dependency relationships to other versions in the partition from which it is being promoted (the source partition).
5. Verify that *element* owns no dependency relationships to other versions in related partitions that will not be visible from the source partition's parent.
6. If the **autopurge** property on the source partition is set, purge previous versions by sending the **purge** message to *element*.
7. Set *element*'s **inPartition** property to the source partition's parent.
8. If the *closure_list* argument was specified, add the promoted version to the list.

Errors

ISRESERVED	*element* is reserved.
NEEDCONTROL	*element* is uncontrolled.
NOBASEPART	The current context has no base partition set.
NOCONTEXT	There is no currently opened context.
NOTVERSIONED	*element* is not a versioned object.
NOTVISIBLE	*element* is not visible in the current context.
NOTMOD	*element* cannot be modified in the current context.
PROFROROO	*element* is in the root partition and cannot be promoted.

purge—Purge Previous Versions

Deletes preceding VERSION elements on the same line of descent as the element to which it is sent. Each element is deleted, provided the following are true:

- The element is not the first on the line of descent.
- The element does not merge into another element or another element does not merge into it.
- The element does not participate in any relationships as member.
- The element is in the same partition as the element to which the message was sent.

Symbolic Name

C Binding: MCS_message_purge
OpenVMS Binding: MCS$r_message_purge

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION **defines**
3 AGGREGATE *inherits*
4 BINARY *inherits*
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE *inherits*

3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*
5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

VERSION

1. Verify that *element* is not reserved.
2. Send the **free** message to each ancestor on *element*'s line of descent, unless:
   a. The ancestor merges into another element.
   b. The ancestor is the first version on the line of descent.
   c. The ancestor is not in the same partition as *element* (in this case do not delete any earlier ancestors either).

rename—Rename Element

Renames the element to which it is sent.

Symbolic Name

C Binding: MCS_message_rename
OpenVMS Binding: MCS$r_message_rename

Required Arguments

MCS_arg_new_name (C binding)
MCS$r_arg_new_name (OpenVMS binding)
data type: MCS_STRING
use: input
New name.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT **defines**
2 CONTEXT *inherits*
2 DATABASE *inherits*
2 DIRECTORY **disallows**
2 PARTITION *inherits*
2 PERSISTENT_PROCESS *inherits*
2 VERSION *inherits*
3 AGGREGATE *inherits*
4 BINARY *inherits*
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE *inherits*

3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*
5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

DIRECTORY

Disallowed.

NAMED_ELEMENT

1. Verify that the argument list contains a value for *element*'s new name.
2. Verify that the new name is not the name of an existing element.
3. Change *element*'s name to the new name.

replace—Replace Reserved Version

"Checks in" the reserved VERSION element to which it is sent. The replaced VERSION element becomes immutable.

Symbolic Name

C Binding: MCS_message_replace
OpenVMS Binding: MCS$r_message_replace

Required Arguments

None.

Optional Arguments

MCS_arg_branch_name (C binding)
MCS$r_arg_branch_name (OpenVMS binding)
data type: MCS_STRING
use: input
Specifies the name of the branch to create for the replaced VERSION element.

- If a branch was specified on the **reserve** message, the branch you supply with **replace** supersedes that branch.
- If you specify a null branch name (a null string), *element* is replaced on its predecessor's line of descent, if possible; otherwise, an error occurs. A null branch name on **replace** cancels a branch specified with **reserve**.

MCS_arg_closure (C binding)
MCS$r_arg_closure (OpenVMS binding)
data type: MCS_SMALLINT
use: input
Indicates which other elements related to the element should be replaced. Values are shown in the following table:

Value	Meaning
MCS_TO_NONE	Replace only this element. This is the default.

Value	Meaning
MCS_TO_TOP	Replace this element and all direct and indirect parents to the top of the composite hierarchy, provided that they do not own (through dependency relationships) other reserved elements. Parents that do own other replaced elements are not replaced, no error message is generated, and their element ID is not placed on the closure list.
MCS_TO_BOTTOM	Replace this element and all elements owned, either directly or indirectly, through dependency relationships. These include direct and indirect children of a composite.
MCS_TO_BOTH	Equivalent to sending **replace** twice, once with MCS_TO_BOTTOM specified, then with MCS_TO_TOP.

MCS_arg_closure_list (C binding)
MCS$r_arg_closure_list (OpenVMS binding)
data type: MCS_LIST
use: output
Receives the list of elements that were replaced.

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
  1 NAMED_ELEMENT *does not recognize*
    2 VERSION **defines**
      3 AGGREGATE *inherits*
        4 BINARY **refines**
          5 BINARY_TOOL *inherits*
          5 TEXT *inherits*
            6 TEXT_TOOL *inherits*
        4 COMPOSITE *inherits*
          5 COLLECTION **refines**
      3 MESSAGE **refines**

3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD **refines**
5 VALIDATION *inherits*
3 TYPE **refines**
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*
5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

BINARY

1. Superop.
2. If *element* is stored externally, terminate the method successfully.
3. Replace the file represented by *element* into the delta file-storage system.
4. Delete the file.

Errors

NOTRESERVED *element* is not reserved.

COLLECTION

1. If *element* is the CDD$METADATA collection and it is being replaced into CDD$METADATA_PARTITION, rebuild the dictionary metadata schema. Following this, the metadata elements contained in the collection become the current dictionary schema.
2. Superop.

MESSAGE

1. Verify that the current context's **top** property is set to the CDD$METADATA collection.
2. If the *closure* argument is not supplied or is supplied as MCS_TO_NONE, set it to MCS_TO_BOTH. In any case, make sure *closure* is set such that the entire metadata collection will be replaced.

3. Superop.

Note

The "metadata collection" is not a true collection in that it is not connected through COMPOSITE_PART subtype relationships. However, this method treats it as a collection.

Errors

BADARGUMENT	An invalid argument was specified.
BADCLOS	An inappropriate value was specified for the *closure* argument.
METATOP	The CDD$METADATA collection is not at the top of the collection hierarchy.

METHOD

See the description of the **replace** method defined by MESSAGE, which is identical to this method.

TYPE

See the description of the **replace** method defined by MESSAGE, which is identical to this method.

VERSION

1. If the *closure* argument was included and its value is MCS_TO_BOTTOM or MCS_TO_BOTH, send the **replace** message with the same *closure* argument value to each of *element*'s children.
2. If *element* is controlled:
   a. Verify that *element* is reserved. (If the method was called recursively to replace all or part of a collection, and *element* is not reserved, the method simply adds *element* to the *closure_list*, if specified, and exits.)
   b. If the *branch_name* argument was not specified or was specified as null, verify that there are no other descendants. Otherwise:
      1. Verify that the value of *branch_name* is a valid branch name.

2. Verify that no other branch exists with this name.

c. Verify that *element* does not own a dependency relationship to another reserved, uncontrolled, or non-versioned element.

d. Set *element*'s **inPartition** property to the current base partition.

e. Set *element*'s **status** property to MCS_STS_AVAIL.

3. If the *closure_list* argument was specified, add the replaced version to the list.

4. If the *closure* argument was included and its value is MCS_TO_TOP or MCS_TO_BOTH, send the **replace** message with the same *closure* argument value to each of *element*'s parents to the top of the collection hierarchy.

Errors

NEEDCONTROL	*element* is uncontrolled.
NOBASEPART	The current context has no base partition defined.
NOCONTEXT	There is no current context.
NOTRESERVED	*element* is not currently reserved.
NOTVERSIONED	*element* is not a versioned element.
NOTVISIBLE	*element* is not visible in the current context.
OWNSRESERVED	*element* owns reserved elements.
TOPNOTSET	The current context does not specify a top composite.

reserve—Reserve Version

"Checks out" a VERSION element to which it is sent by creating a new VERSION element. If you supply the *branch_name* argument, the message creates a new branch and makes the new version the first on that branch. The message returns the element ID of the created element in the *new_inst* argument.

Symbolic Name

C Binding: MCS_message_reserve
OpenVMS Binding: MCS$r_message_reserve

Required Arguments

MCS_arg_new_inst_elmID (C binding)
MCS$r_arg_new_inst_elmID (OpenVMS binding)
data type: MCS_ELEMENTID
use: output
Element ID of the reserved VERSION element created by the operation.

Optional Arguments

MCS_arg_branch_name (C binding)
MCS$r_arg_branch_name (OpenVMS binding)
data type: MCS_STRING
use: input
Name of the branch to create. If you omit this argument and the element to which you send the message is not available to be reserved, the operation will fail.

MCS_arg_closure (C binding)
MCS$r_arg_closure (OpenVMS binding)
data type: MCS_SMALLINT
use: input
Indicates whether other elements related to the element should be reserved.

Values are shown in the following table:

Value	Meaning
MCS_TO_NONE	Reserve only this element. This is the default.
MCS_TO_TOP	Reserve this element and all direct and indirect parents to the top of the composite hierarchy.
MCS_TO_BOTTOM	Reserve this element and all direct and indirect children in the composite hierarchy.
MCS_TO_BOTH	Reserve this element and all direct and indirect parents to the top of the composite hierarchy, and all direct and indirect children in the composite hierarchy.
MCS_TO_CLOSURE	Reserve this element and all elements that own it, either directly or indirectly, through dependency relationships and that lie between this element and the top of the composite hierarchy. These elements include the direct and indirect parents of the reserved element, but also include non-composite elements in the composite hierarchy that depend on the reserved element. Thus, the elements reserved by MCS_TO_CLOSURE are a superset of those reserved by MCS_TO_TOP.

MCS_arg_closure_list (C binding)
MCS$r_arg_closure_list (OpenVMS binding)
data type: MCS_LIST
use: output
Receives the list of elements created during the operation; particularly useful in conjunction with the *closure* argument.

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

reserve

Method Summary

0 ELEMENT *does not recognize*
  1 NAMED_ELEMENT *does not recognize*
    2 VERSION **defines**
      3 AGGREGATE *inherits*
        4 BINARY **refines**
          5 BINARY_TOOL *inherits*
          5 TEXT *inherits*
            6 TEXT_TOOL *inherits*
        4 COMPOSITE *inherits*
          5 COLLECTION **refines**
      3 MESSAGE **refines**
      3 MSGARG *inherits*
      3 TOOL *inherits*
        4 METHOD **refines**
          5 VALIDATION *inherits*
      3 TYPE **refines**
        4 DATA_TYPE *inherits*
        4 ELEMENT_TYPE *inherits*
          5 RELATION_TYPE *inherits*
        4 PROPERTY_TYPE *inherits*

Methods

BINARY

1. If *element* is open, send the **close** message to it.
2. Superop.
3. If *element* is stored externally, terminate the method successfully.
4. Fetch the file into the subdirectory of the context directory whose name is identical to the Oracle CDD/Repository directory. The name of the file is the simple name of the element.

COLLECTION

1. Superop.
2. If *element* is the value of the **currCollection** property in the current persistent process, make the new collection the value of the **currCollection** property.

MESSAGE

1. Verify that the current context's **top** property is set to the CDD$METADATA collection.
2. Verify that the *closure* argument (if supplied) is not set to MCS_TO_BOTTOM. If *closure* is not supplied or is supplied as MCS_TO_NONE, set it to MCS_TO_TOP.
3. Verify that the *branch_name* argument has not been supplied or has been supplied as NULL.
4. Superop.

Note

The "metadata collection" is not a true collection in that it is not connected through COMPOSITE_PART subtype relationships. However, this method treats it as a collection.

Errors

BADARGUMENT	An invalid argument was specified.
BADCLOS	The value supplied for the *closure* argument is not supported for this operation.
METATOP	The CDD$METADATA collection is not at the top of the collection hierarchy.
NOBRMETA	An attempt was made to reserve *element* on a branch. Metadata elements must be reserved on the main line of descent.

METHOD

See the description of the **reserve** method defined by MESSAGE, which is identical to this method.

TYPE

See the description of the **reserve** method defined by MESSAGE, which is identical to this method.

VERSION

1. If there is a current context, verify that *element* resides in a partition that lies in the path from the context to the root of the partition hierarchy.
2. Verify that *element* is not reserved. (If the method was called recursively to reserve collections above the original target of the message, and *element* is reserved, the method simply exits.)
3. If the *closure* argument was included and its value is MCS_TO_TOP or MCS_TO_BOTH, send the **reserve** message with the same *closure* argument value to each of *element*'s parents to the top of the collection hierarchy.
4. Set *element*'s **status** property to MCS_STS_RO (read-only).
5. Create a new version of *element*.
6. Copy the following properties and relationships from *element* to its new version:
   - all normal properties
   - all required relation properties
   - any relationships whose type is COMPOSITE_PART or any of its subtypes
7. If a branch name was specified with the message:
   a. Verify that the branch name is valid.
   b. Verify that a current context exists.
8. Make the new version a descendant of *element*.
9. If *element* is not at the top of the composite hierarchy, then for each reserved parent in context of *element*, change the COMPOSITE_PART relationship to point to the newly created version. There must be at least one reserved parent in context; if there is none, an error results. (This implies that *element* must be either the top of the composite hierarchy or under the top in order for it to be reserved.)

   Otherwise, *element* is the top of the composite hierarchy. Set the value of **top** to the newly created element.
10. If the *closure_list* argument was specified, add the new version to the list.
11. If the *closure* argument was included and its value is MCS_TO_BOTTOM or MCS_TO_BOTH, send the **reserve** message with the same *closure* argument value to each of *element*'s children.

Errors

BADARGUMENT	An invalid argument was included in the message argument list.
ERRDESCENDEXIST	*element* already has a descendent on the same line of descent.
FROZEN	*element* is frozen.
ISRESERVED	*element* is already reserved.
NEEDCONTROL	*element* is uncontrolled.
NOBASEPART	The current context has no base partition defined.
NOCONTEXT	No context has been opened.
NOTMOD	*element* cannot be modified in this context.
NOTUNDERTOP	*element* is not in the composite hierarchy headed by the current top composite.
NOTVERSIONED	*element* is not a versioned object.
NOTVISIBLE	*element* is not visible in this context.
PARNOTRESERVED	There is no reserved parent in this context.
TOPNOTSET	The current context does not identify a top composite.

setProp—Set Property Values

Sets the value of one or more properties belonging to the element to which it is sent. For all properties except **history**, **setProp** also sends notifications to owners of dependency relationships of which the receiving element is a member.

Symbolic Name

C Binding: MCS_message_setProp
OpenVMS Binding: MCS$r_message_setProp

Required Arguments

MCS_arg_arglist (C binding)
MCS$r_arg_arglist (OpenVMS binding)
data type: MCS_LIST
use: input-output
List of properties to be set. Each element of the argument list contains three items:

- Property name—The name of the property to set.
- Property value—The value to which to set the property.
- Status—Oracle CDD/Repository returns a code indicating whether the property was successfully set.

  If you set the status to MCS_MISSING for a particular property before sending the **setProp** message, the property value is removed; the effect is as if it had never been set. If the property is a relation property, the entire scan is deleted: all relationships that implement the scan are deleted, but the elements in the scan are not deleted.

Use the various MCS_arglist routines to access the elements of an argument list.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT **defines**
- 1 EVENT *inherits*
- 1 METHOD_INVOCATION *inherits*
  - 2 ACAS_METHOD_INVOC *inherits*
  - 2 ATIS_METHOD_INVOC *inherits*
- 1 NAMED_ELEMENT *inherits*
  - 2 CONTEXT **refines**
  - 2 DATABASE *inherits*
  - 2 DIRECTORY *inherits*
  - 2 PARTITION **refines**
  - 2 PERSISTENT_PROCESS **refines**
  - 2 VERSION **refines**
    - 3 AGGREGATE *inherits*
      - 4 BINARY *inherits*
        - 5 BINARY_TOOL *inherits*
        - 5 TEXT *inherits*
          - 6 TEXT_TOOL *inherits*
      - 4 COMPOSITE *inherits*
        - 5 COLLECTION *inherits*
    - 3 MESSAGE *inherits*
    - 3 MSGARG *inherits*
    - 3 TOOL *inherits*
      - 4 METHOD *inherits*
        - 5 VALIDATION *inherits*
    - 3 TYPE *inherits*
      - 4 DATA_TYPE *inherits*
      - 4 ELEMENT_TYPE *inherits*
        - 5 RELATION_TYPE *inherits*
      - 4 PROPERTY_TYPE *inherits*
- 1 RELATION *inherits*
  - 2 DEPENDS_ON **refines**
    - 3 COMPOSITE_PART *inherits*
    - 3 HAS_DEFAULT_METHOD *inherits*
    - 3 HAS_MESSAGE *inherits*
    - 3 HAS_MSGARG *inherits*
    - 3 HAS_PROPERTY *inherits*
      - 4 HAS_COMPUTED_PROPERTY *inherits*
      - 4 HAS_RELATION_PROPERTY *inherits*
    - 3 HAS_RELATION *inherits*
    - 3 HAS_SUPERTYPE *inherits*
    - 3 IMPLEMENTS_METHOD *inherits*

3 IMPLEMENTS_RELATION *inherits*
3 INVOKES_TOOL *inherits*
3 RELATION_MEMBER *inherits*
2 HAS_CONTEXT *inherits*
2 HAS_CURR_COLLECTION *inherits*
2 HAS_DATATYPE *inherits*
2 HAS_PARENT *inherits*
2 HAS_POSTAMBLE *inherits*
2 HAS_PREAMBLE *inherits*
2 HAS_RELATED_PARTITION *inherits*
2 HAS_TOP_COLLECTION *inherits*
2 OPENED_BY *inherits*
2 RESERVED_BY *inherits*

Methods

CONTEXT

For **contextDir**:

1. Verify that the specified value is not null.
2. Rename the context directory to the new name.
3. Superop.

For **top**:

1. Verify that *element* owns no reserved elements or open files.
2. Recursively delete all directories under the context directory.
3. Superop.

Errors

CERROR	A C run-time error occurred.
SETPROPFAILED	One or more of the specified properties could not be set.

DEPENDS_ON

1. Verify that each versionable owner of *element* is reserved.
2. If the property being changed is **relMember**, send a notification to all owners of *element*.
3. Superop.

ELEMENT

1. Verify that *element* possesses the property being set.
2. Verify that the access type of the property allows it to be set. The property must be one of the following:
   - Read/write
   - Write-once, and not previously written
   - Write-once-at-creation, in which case the **setProp** message must have been dispatched by a **new** method
3. Set the value of the property to its new value.
4. For properties other than **history**, send a notification to each owner of a dependence relationship for which *element* is a member.

Errors

ERRSETPROP	An attempt was made to set the value of a computed property.
NOTAPPLICABLE	An attempt was made to set the value of a property that *element* does not possess.
SETPROPFAILED	One or more of the specified properties could not be set.

PARTITION

For **partitionDir**:

1. Verify that there are no open files in the current partition.
2. Verify that the supplied value is not null.
3. Superop.
4. Rename *element*'s partition directory to the new directory name.

Errors

BADOPENDIR	An error occurred while trying to open a directory.
BADREADDIR	An error occurred while trying to read a directory.
DIR_BADNAM	An error occurred while trying to parse a name.
SETPROPFAILED	One or more of the specified properties could not be set.

PERSISTENT_PROCESS

For **currContext**:

1. Send the **close** message to *element*'s current context, if one exists.
2. Superop.
3. Send the **open** message to *element*'s new current context.

VERSION

1. If *element* is controlled and the property to be set is an "immutable" property, verify that *element* is reserved. (An immutable property is one that cannot be set unless the version is reserved.)
2. Superop.

translate—Create New Derived Version

Creates a new version of an element derived from this element if this element has changed since the last time it was translated.

Note

The **translate** message performs no action in this release of Oracle CDD/Repository. It acts as a placeholder for future releases.

Symbolic Name

C Binding: MCS_message_translate
OpenVMS Binding: MCS$r_message_translate

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION **defines**
3 AGGREGATE *inherits*
4 BINARY *inherits*
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE *inherits*
3 MSGARG *inherits*
3 TOOL *inherits*

4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*
5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

VERSION

For this version of Oracle CDD/Repository, this is a null method.

unfreeze—Unfreeze Version

Reverses a previous **freeze** message; it allows the VERSION element to which it is sent to be reserved.

Symbolic Name

C Binding: MCS_message_unfreeze
OpenVMS Binding: MCS$r_message_unfreeze

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION **defines**
3 AGGREGATE *inherits*
4 BINARY *inherits*
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE *inherits*
3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*

5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*

Methods

VERSION

1. Verify that *element* is frozen.
2. If *element* is the available version (the last checked-in version) on its line of descent, set its **status** to MCS_STS_AVAIL; otherwise, set **status** to MCS_STS_RO.

Errors

NOCONTEXT	There is no currently open context.
NOBASEPART	The current context has no base partition set.
NOTFROZEN	*element* is not frozen.
NOTVERSIONED	*element* is not a versioned element.

unreserve—Unreserve Reserved Version

Sent to a reserved element, cancels a previous **reserve** operation by deleting the element. The message returns the predecessor of the deleted element in the *old_inst* argument.

Symbolic Name

C Binding: MCS_message_unreserve
OpenVMS Binding: MCS$r_message_unreserve

Required Arguments

MCS_arg_old_inst_elmID (C binding)
MCS$r_arg_old_inst_elmID (OpenVMS binding)
data type: MCS_ELEMENT_ID
use: output
Element ID of the unreserved element's predecessor.

Optional Arguments

MCS_arg_closure (C binding)
MCS$r_arg_closure (OpenVMS binding)
data type: MCS_SMALLINT
use: input
Indicates how much of the composite containing the element should be unreserved. Values are shown in the following table:

Value	Meaning
MCS_TO_NONE	Unreserve only this element. This is the default.
MCS_TO_TOP	Unreserve this element and all direct and indirect parents to the top of the composite hierarchy.
MCS_TO_BOTTOM	Unreserve this element and all direct and indirect children in the composite hierarchy.

Value	Meaning
MCS_TO_BOTH	Unreserve this element and all direct and indirect parents to the top of the composite hierarchy, and all direct and indirect children in the composite hierarchy.

MCS_arg_closure_list (C binding)
MCS$r_arg_closure_list (OpenVMS binding)
data type: MCS_LIST
use: output
Receives the list of elements that were unreserved.

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
  1 NAMED_ELEMENT *does not recognize*
    2 VERSION **defines**
      3 AGGREGATE *inherits*
        4 BINARY **refines**
          5 BINARY_TOOL *inherits*
          5 TEXT *inherits*
            6 TEXT_TOOL *inherits*
        4 COMPOSITE *inherits*
          5 COLLECTION **refines**
      3 MESSAGE **disallows**
      3 MSGARG *inherits*
      3 TOOL *inherits*
        4 METHOD **disallows**
      3 TYPE **disallows**

Methods

BINARY

1. If *element* is stored internally, delete the file represented by *element*.
2. Superop.

Errors

NOTRESERVED *element* is not reserved.

COLLECTION

1. Superop.
2. If *element* was the value of **currCollection**, change the value of **currCollection** to *element*'s predecessor.
3. If opened files no longer have a parent in context, close them.

MESSAGE

Disallowed. Metadata elements cannot be unreserved.

METHOD

Disallowed. Metadata elements cannot be unreserved.

TYPE

Disallowed. Metadata elements cannot be unreserved.

VERSION

1. Verify that *element* resides in a partition.
2. Verify that *element* is reserved in this context. (If the method was called recursively to unreserve all or part of a collection, and *element* is not reserved, the method simply adds *element* to the *closure_list*, if specified, and exits.)
3. If the *closure* argument was included and its value is MCS_TO_BOTTOM or MCS_TO_BOTH, send the **unreserve** message with the same *closure* argument value to each of *element*'s children.
4. Set *element*'s predecessor's **status** property to MCS_STS_AVAIL.
5. If *element* is not the top of the current collection hierarchy, then for each of *element*'s parents in context, change the COLLECTION_PART relationship to point to the predecessor. Otherwise, set the top of the collection hierarchy to the predecessor.

6. Delete all the COLLECTION_PART relationships of which *element* is an owner or member.
7. Delete *element* itself.
8. If the *closure_list* argument was specified, add the predecessor version to the list.
9. If the *closure* argument was included and its value is MCS_TO_TOP or MCS_TO_BOTH, send the **unreserve** message with the same *closure* argument value to each of *element*'s parents to the top of the collection hierarchy.

Errors

BADCLOS	An inappropriate value was specified for the *closure* argument.
ISRESERVED	*element* contains one or more reserved children, which must be unreserved before *element* can be unreserved.
NEEDCONTROL	*element* is not a controlled element.
NOBASEPART	The current context does not identify a base partition.
NOCONTEXT	No context has been opened.
NOTRESERVED	*element* is not reserved in this context.
NOTUNDERTOP	*element* is not in the composite hierarchy identified by the current top composite.
NOTVERSIONED	*element* is not a versioned object.
NOTVISIBLE	*element* is not visible in the current context.
OWNSRESERVED	*element* owns reserved versions.
TOPNOTSET	The current context does not identify a composite hierarchy.

update—Update Composite

Sent to a COMPOSITE element, changes the versions attached to the composite. The update action for each child depends on the following values:

- the value of the **attachment** property on the COMPOSITE_PART relationship that attaches the child to the composite
- if **attachment** is not set, the value of **defaultAttachment** on the current context
- if neither **attachment** nor **defaultAttachment** is set, the value MCS_ATTACH_LATEST

Depending on this value, **update** performs the following action:

- If the value is MCS_ATTACH_SPEC_VERSION, do not change the attached version attached to the composite.
- If the value is MCS_ATTACH_LAST_CHKIN, find the latest replaced version of the element and replace the previously attached version with it.
- If the value is MCS_ATTACH_LATEST, find the latest version of the element and replace the previously attached version with it.

Symbolic Name

C Binding: MCS_message_update
OpenVMS Binding: MCS$r_message_update

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

Method Summary

0 ELEMENT *does not recognize*
1 NAMED_ELEMENT *does not recognize*
2 VERSION *does not recognize*
3 AGGREGATE *does not recognize*
4 COMPOSITE **defines**
5 COLLECTION *inherits*

Methods

COMPOSITE

1. Verify that *element* is reserved.
2. For each of *element*'s unreserved non-composite children, find the attachment value (see the general description of the **update** message) and (if necessary) detach the child and attach the appropriate version to *element*. Note that a reserved child is never detached.

Errors

BADPARAM	An illegal parameter was supplied.
NOTRESERVED	*element* is not reserved.

verify—Verify Element

Verifies the integrity of the element to which it is sent. The *crash_level* argument determines the extent of verification to be done. Oracle CDD/Repository currently defines the following symbolic values for *crash_level*:

- MCS_VERIFY_INTERNAL_STRUCTURE

  Check the internal structure of the element. For example, check that the values of all properties are of the correct data types. This usually means running Oracle CDD/Repository validations.

- MCS_VERIFY_REPAIR

  Perform structural checks, then modify the element to fix any discrepancies found. Note that the extent to which automatic repair is possible will vary depending on the element type and the discrepancy. Many problems will not be repairable without manual intervention.

It is the responsibility of the **verify** method refinement on each subtype of ELEMENT to verify or repair those properties and semantics that are defined at that level in the type hierarchy. The refinement should dispatch the superop at the appropriate point to cause verification of those properties defined at higher levels.

Symbolic Name

C Binding: MCS_message_verify
OpenVMS Binding: MCS$r_message_verify

Required Arguments

None.

Optional Arguments

MCS_arg_comment (C binding)
MCS$r_arg_comment (OpenVMS binding)
data type: MCS_LIST
use: input
List of strings to form a comment for this operation.

MCS_arg_crash_level (C binding)
MCS$r_arg_crash_level (OpenVMS binding)
data type: MCS_LONGINT
use: input
Verification operation to be performed. Values are shown in the following table:

Value	Meaning
MCS_VERIFY_INTERNAL_ STRUCTURE	Check internal structure.
MCS_VERIFY_REPAIR	Repair the object.

The default is MCS_VERIFY_INTERNAL_STRUCTURE.

Method Summary

0 ELEMENT **defines**
1 EVENT *inherits*
1 METHOD_INVOCATION *inherits*
2 ACAS_METHOD_INVOC *inherits*
2 ATIS_METHOD_INVOC *inherits*
1 NAMED_ELEMENT *inherits*
2 CONTEXT **refines**
2 DATABASE *inherits*
2 DIRECTORY *inherits*
2 PARTITION *inherits*
2 PERSISTENT_PROCESS *inherits*
2 VERSION *inherits*
3 AGGREGATE *inherits*
4 BINARY **refines**
5 BINARY_TOOL *inherits*
5 TEXT *inherits*
6 TEXT_TOOL *inherits*
4 COMPOSITE *inherits*
5 COLLECTION *inherits*
3 MESSAGE *inherits*
3 MSGARG *inherits*
3 TOOL *inherits*
4 METHOD *inherits*
5 VALIDATION *inherits*
3 TYPE *inherits*
4 DATA_TYPE *inherits*
4 ELEMENT_TYPE *inherits*

5 RELATION_TYPE *inherits*
4 PROPERTY_TYPE *inherits*
1 RELATION *inherits*
2 DEPENDS_ON *inherits*
3 COMPOSITE_PART *inherits*
3 HAS_DEFAULT_METHOD *inherits*
3 HAS_MESSAGE *inherits*
3 HAS_MSGARG *inherits*
3 HAS_PROPERTY *inherits*
4 HAS_COMPUTED_PROPERTY *inherits*
4 HAS_RELATION_PROPERTY *inherits*
3 HAS_RELATION *inherits*
3 HAS_SUPERTYPE *inherits*
3 IMPLEMENTS_METHOD *inherits*
3 IMPLEMENTS_RELATION *inherits*
3 INVOKES_TOOL *inherits*
3 RELATION_MEMBER *inherits*
2 HAS_CONTEXT *inherits*
2 HAS_CURR_COLLECTION *inherits*
2 HAS_DATATYPE *inherits*
2 HAS_PARENT *inherits*
2 HAS_POSTAMBLE *inherits*
2 HAS_PREAMBLE *inherits*
2 HAS_RELATED_PARTITION *inherits*
2 HAS_TOP_COLLECTION *inherits*
2 OPENED_BY *inherits*
2 RESERVED_BY *inherits*

Methods

BINARY

1. Superop.
2. Call the delta file mechanism to verify the integrity of the delta file indicated by *element*'s **deltaFile** property.

Errors Various DELTA status codes.

CONTEXT

1. If the value of the *crash_level* argument is REPAIR:
   a. Create the context directory if it does not already exist.

b. Determine whether reserved files and the directories that contain them exist in the file system; if they do not, fetch them from the delta file mechanism.

c. Determine whether open files and the partition directories that contain them exist in the file system. If they do not, fetch them from the delta file mechanism.

2. Superop.

Errors

BADMKDIR	An error occurred while trying to create the context directory.

ELEMENT

1. Call Oracle CDD/Repository validations to verify the structure of the element.
2. If the *crash_level* argument is MCS_VERIFY_INTERNAL_STRUCTURE, terminate the method since there is no structure to verify for ELEMENT.
3. Verify that the **elementType** property identifies an element of type ELEMENT_TYPE.
4. Call any Oracle CDD/Repository element repair functions that apply.

3
Property Descriptions

This chapter contains descriptions of all the Oracle CDD/Repository properties, arranged alphabetically. Each property description contains the following sections:

Title—Includes the generic name of the property, a short phrase that describes the purpose, and a paragraph briefly describing use.

Symbolic Name—Gives the symbolic name of the property. Use the symbol to identify the property in argument lists.

Tag—For normal properties, gives the symbolic constant for the value of the **tag** property for this property definition.

Defined By—Gives the name of the element type that defines the property. Subtypes of that element type inherit the property's definition.

If more than one element type is listed, the element types are on separate branches of the type hierarchy.

Required With new—Specifies whether you must specify an initial value for this property when you send the **new** message to create an instance of its defining type or a subtype.

Type—Gives the property's implementation type, as follows:

- Normal—The data is stored with the element.
- Relation—The property's value (an element ID or scan of element IDs) is formed by traversing relationships, and modified by adding or removing relationships.
- Closure—Similar to relation, except that the traversal is recursive.
- Computed—The property's value is computed by a method.

Data Type—Gives the name of the property's data type.

Access—Specifies the allowable access to the property using **getProp** and **setProp**. The values are as follows:

- Read-only—The property may only be read.
- Read/write—The property may be read and its value modified.
- Write-once—The property may be read and its value may be set once.
- Write-once-at-creation—The property may be read and its value may be set when the element that possesses it is created.

The access described here is for all users and is in addition to the access control provided for elements on a user-by-user basis. Note that the value of a property that cannot be changed with **setProp** may still be modified as the side effect of some other operation.

Constant Values—Applies only to integer-valued properties whose values come from a set of constants, or to properties whose values are lists of such integers. Lists the symbolic names for the values that the property may assume and the meaning of each name. The C binding symbol for the value is shown. To form the OpenVMS binding symbol, you must preface the symbol with MCS$K_ instead of MCS_. For example, MCS_TRUE becomes MCS$K_TRUE.

Relation Traversed—Applies only to relation and closure properties. Names the relation that implements the property.

Traversal Direction—Applies only to relation and closure properties. Specifies the direction in which the implementing relation is traversed, either To owner or To member. For closure properties, a value of To all owners or To all members indicates the recursive nature of the traversal.

Inverse Property—Applies only to relation and closure properties. Names the property that traverses the same relation in the opposite direction.

access—Access

Lists access control entries that make up the element access control list for this element.

Summary

Symbolic Name	
C binding	MCS_prop_access
OpenVMS binding	MCS$r_prop_access
Tag	NAD$K_ATT_ACL
Defined By	NAMED_ELEMENT
Required With **new**	No
Type	Normal
Data Type	MCS_MEMBLOCK
Access	Read/write

Description

The value of this property is an access control list. When you establish the value with the **new** or **setProp** message, you insert a complete binary access control list buffer into the property value. When you use **getProp** to read the property value, Oracle CDD/Repository inserts a complete binary access control list buffer into the value structure.

The data type of this property is MCS_MEMBLOCK, an unstructured block of memory. Its format conforms to the binary format for access control lists described in the OpenVMS documentation about system services. The section on security services in the OpenVMS documentation about system services provides information on how to translate text into binary and how to translate binary into text.

Each privilege mask is a longword. Bits 0 through 3 and 16 through 19 describe privileges that apply to element types; bits 4 through 8 and 20 through 24 describe privileges that apply to element instances. Bits 15 and 31 describe privileges that apply to all elements. Bits 9 through 14 and 25 through 30 are reserved for future use.

access

Table 3–1 lists the Oracle CDD/Repository privileges and the bits with which they are associated.

Table 3–1 Privilege Bits for Access Control Lists

Privilege	Bit
Element Type Privileges	
READ	0
WRITE	1
MODIFY	2
ERASE	3
NOREAD	16
NOWRITE	17
NOMODIFY	18
NOERASE	19
Element Instance Privileges	
SHOW	4
DEFINE	5
CHANGE	6
DELETE	7
CONTROL	8
NOSHOW	20
NODEFINE	21
NOCHANGE	22
NODELETE	23
NOCONTROL	24
Privileges for All Elements	
ALL	15
NOALL	31

See *Using Oracle CDD/Repository on OpenVMS Systems* for information about these privileges.

accessType—Access Type

Indicates how a property may have its value set.

Summary

Symbolic Name	
C binding	MCS_prop_accessType
OpenVMS binding	MCS$r_prop_accessType
Tag	NAD$K_ATT_ACCESS
Defined By	ELEMENT_TYPE
Required With **new**	Yes
Type	Normal
Data Type	MCS_SMALLINT
Access	Write-once-at-creation

Constant Values

Value	Meaning
MCS_PROP_ACCESS_READONLY	Read-only
MCS_PROP_ACCESS_READWRITE	Read/write
MCS_PROP_ACCESS_WRITEONCE	Write-once
MCS_PROP_ACCESS_WRITECREATE	Write-once-at-creation

aliases—Command Aliases

Lists strings in the form *alias-name=value*. This property supports DCL global symbols.

aliases

Summary

Symbolic Name	
C binding	MCS_prop_aliases
OpenVMS binding	MCS$r_prop_aliases
Tag	NAD$K_ATT_ALIASES
Defined By	PERSISTENT_PROCESS
Required With **new**	No
Type	Normal
Data Type	MCS_LIST
Access	Read/write

allCheckouts—All Reserved Versions

Identifies all the VERSION elements that are reserved in the repository represented by this DATABASE element.

Summary

Symbolic Name	
C binding	MCS_prop_allCheckouts
OpenVMS binding	MCS$r_prop_allCheckouts
Defined By	DATABASE
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

allChildPartitions—All Child Partitions

Identifies all the PARTITION elements that are children of the current PARTITION, all their children, and so on. The value of this property is established and modified when new PARTITION elements are created that specify the PARTITION element possessing the property (or one of its children) as parent.

Summary

Symbolic Name	
C binding	MCS_prop_allChildPartitions
OpenVMS binding	MCS$r_prop_allChildPartitions
Defined By	PARTITION
Required With **new**	No
Type	Closure
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	HAS_PARENT
Traversal Direction	To all owners
Inverse Property	**allParentPartitions**

allChildren—All Composite Children

Identifies all the VERSION elements contained by this composite and all the composites that it contains. The value of this property is modified as the result of **attach** and **detach** messages.

allChildren

Summary

Symbolic Name	
C binding	MCS_prop_allChildren
OpenVMS binding	MCS$r_prop_allChildren
Defined By	COMPOSITE
Required With **new**	No
Type	Closure
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	COMPOSITE_PART
Traversal Direction	To all members
Inverse Property	None

allDependencies—All Dependencies

Identifies all the elements that depend on the element possessing the property.

Summary

Symbolic Name	
C binding	MCS_prop_allDependencies
OpenVMS binding	MCS$r_prop_allDependencies
Defined By	VERSION
Required With **new**	No
Type	Closure
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	DEPENDS_ON
Traversal Direction	To all owners
Inverse Property	**allDependents**

allDependents—All Dependents

Identifies all the elements on which the element possessing the property depends.

Summary

Symbolic Name	
C binding	MCS_prop_allDependents
OpenVMS binding	MCS$r_prop_allDependents
Defined By	VERSION
Required With **new**	No
Type	Closure
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	DEPENDS_ON
Traversal Direction	To all members
Inverse Property	**allDependencies**

allDerivedFrom—All Versions Derived From

Identifies the VERSION elements from which this version derives, both directly and indirectly. The scan includes METHOD_INVOCATION elements that record the process of creating this version.

allDerivedFrom

Summary

Symbolic Name	
C binding	MCS_prop_allDerivedFrom
OpenVMS binding	MCS$r_prop_allDerivedFrom
Defined By	VERSION
Required With **new**	No
Type	Closure
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	METHOD_PARAMETER
Traversal Direction	To all members
Inverse Property	**allDerives**

allDerives—All Versions Deriving from This Version

Identifies the VERSION elements deriving from this version, both directly and indirectly. The scan includes METHOD_INVOCATION elements that record the process by which the result versions were created.

Summary

Symbolic Name	
C binding	MCS_prop_allDerives
OpenVMS binding	MCS$r_prop_allDerives
Defined By	VERSION
Required With **new**	No
Type	Closure
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	METHOD_PARAMETER
Traversal Direction	To all owners
Inverse Property	**allDerivedFrom**

allElementTypes—All Element Types

Identifies element types of the element possessing the property. The scan includes the element type of the element and all its supertypes. The value of this property is established when an element type is created and its supertype specified.

Summary

Symbolic Name	
C binding	MCS_prop_allElementTypes
OpenVMS binding	MCS$r_prop_allElementTypes
Defined By	ELEMENT
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

allInstances—All Instances of Type

Identifies the instances of this type and its subtypes. The value of this property changes as instances of the type and its subtypes are created and deleted.

Summary

Symbolic Name	
C binding	MCS_prop_allInstances
OpenVMS binding	MCS$r_prop_allInstances
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

allParentPartitions—All Parent Partitions

Identifies the parent PARTITION, its parent, and so on. The value of this property is established when a PARTITION element, at the time of its creation, specifies a parent partition.

Summary

Symbolic Name	
C binding	MCS_prop_allParentPartitions
OpenVMS binding	MCS$r_prop_allParentPartitions
Defined By	PARTITION
Required With **new**	No
Type	Closure
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	HAS_PARENT
Traversal Direction	To all members
Inverse Property	**allChildPartitions**

allSubTypes—All Subtypes

Identifies the immediate subtypes of a type, any subtypes they may have, and so on.

Summary

Symbolic Name	
C binding	MCS_prop_allSubTypes
OpenVMS binding	MCS$r_prop_allSubTypes
Defined By	TYPE
Required With **new**	No
Type	Closure
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	HAS_SUPERTYPE
Traversal Direction	To all owners
Inverse Property	**allSuperTypes**

allSuperTypes—All Supertypes

Identifies the immediate supertype of a type, any supertype it may have, and so on.

Summary

Symbolic Name	
C binding	MCS_prop_allSuperTypes
OpenVMS binding	MCS$r_prop_allSuperTypes
Tag	
Defined By	TYPE
Required With **new**	No
Type	Closure
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	HAS_SUPERTYPE
Traversal Direction	To all members
Inverse Property	**allSubTypes**

alternateNames—Alternate Element Names

Lists alternate names for the element. The first name in the list is the value of the element's **name** property; changing this name also changes the value of the **name** property.

Summary

Symbolic Name	
C binding	MCS_prop_alternateNames
OpenVMS binding	MCS$r_prop_alternateNames
Tag	NAD$K_ATT_NAD_DIRNAME
Defined By	NAMED_ELEMENT
Required With **new**	No
Type	Normal
Data Type	MCS_LIST
Access	Read/write

application—Application

Names the application that implements a method.

Summary

Symbolic Name	
C binding	MCS_prop_application
OpenVMS binding	MCS$r_prop_application
Tag	NAD$K_ATT_METH_APPL
Defined By	METHOD
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Read/write

argList—Argument List

Synonym for the **argSpec** property.

argSpec—Argument Specification

Identifies MSGARG elements that describe the functional interface provided by a message.

Summary

Symbolic Name	
C binding	MCS_prop_argSpec
OpenVMS binding	MCS$r_prop_argSpec
Defined By	MESSAGE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_MSGARG
Traversal Direction	To member
Inverse Property	**messagesHavingMsgarg**

argsSent—Method Argument List

Lists the arguments sent with the message that resulted in creation of an ATIS_METHOD_INVOCATION element. Each argument is represented by a string of the form *argument_name=argument_value*.

argsSent

Summary

Symbolic Name	
C binding	MCS_prop_argsSent
OpenVMS binding	MCS$r_prop_argsSent
Tag	NAD$K_ATT_ARGSSENT
Defined By	ATIS_METHOD_INVOC
Required With **new**	No
Type	Normal
Data Type	MCS_LIST
Access	Write-once-at-creation

---

associatedValidations—Associated Validations

Identifies VALIDATION elements that operate on an element type.

Summary

Symbolic Name	
C binding	MCS_prop_associatedValidations
OpenVMS binding	MCS$r_prop_associatedValidations
Defined By	MESSAGE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	OBJECT_VALIDATION
Traversal Direction	To member
Inverse Property	**validationForType**

attachment—Composite Attachment Type

Identifies the attachment type for a member of a composite.

Summary

Symbolic Name	
C binding	MCS_prop_attachment
OpenVMS binding	MCS$r_prop_attachment
Tag	NAD$K_ATT_ATTACHMENT
Defined By	COMPOSITE_PART
Required With **new**	No
Type	Normal
Data Type	MCS_SMALLINT
Access	Read/write

Constant Values

Value	Meaning
MCS_ATTACH_SPEC_VERSION	Attach specific version.
MCS_ATTACH_LAST_CHKIN	Attach last checked-in version.
MCS_ATTACH_LATEST	Attach reserved or last checked-in version.

The values of the **attachment** property are mutually exclusive.

attachmentInContext—Composite Attachment Type in Context

Lists attachments of a VERSION element to its parent composite(s) in the current context. If you specify a composite in the **getProp** argument list (by including the *collection_elmID* argument), the value of this property is the attachment of the version to that composite. If you do not specify a composite, the list contains the attachments of the version to each parent-in-context. In either case, if no attachment is found for a particular composite, the value of the current context's **defaultAttachment** property is substituted.

attachmentInContext

Summary

Symbolic Name	
C binding	MCS_prop_attachmentInContext
OpenVMS binding	MCS$r_prop_attachmentInContext
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_LIST
Access	Read-only

Constant Values

Value	Meaning
MCS_ATTACH_SPEC_VERSION	Attach specific version.
MCS_ATTACH_LAST_CHKIN	Always use last checked in version.
MCS_ATTACH_LATEST	Reserved or last checked in version.

The values of the **attachmentInContext** property are mutually exclusive.

autopurge—Automatic Partition Purge

Indicates whether intermediate versions of elements in a partition should be purged when the latest version is promoted from that partition.

Summary

Symbolic Name	
C binding	MCS_prop_autopurge
OpenVMS binding	MCS$r_prop_autopurge
Tag	NAD$K_ATT_PURGE_V
Defined By	PARTITION
Required With **new**	No
Type	Normal
Data Type	MCS_BOOLEAN
Access	Read/write

Constant Values

Value	Meaning
TRUE	Intermediate versions should be purged.
FALSE	Intermediate versions should not be purged.

availVersion—Available Version

Identifies the most recently replaced version on the line of descent containing the version to which the **getProp** message is sent. This is the version that is available for reading, subject to access control. For a reserved version that is the only version on its line of descent, the property is null since there is no available version. The value of this property changes when a user replaces a reserved version.

availVersion

Summary

Symbolic Name	
C binding	MCS_prop_availVersion
OpenVMS binding	MCS$r_prop_availVersion
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_ELEMENTID
Access	Read-only

---

basePartition—Base Partition

Identifies the partition from which searches of the partition hierarchy start. Searches start from the base partition and proceed to the root of the partition hierarchy.

Summary

Symbolic Name	
C binding	MCS_prop_basePartition
OpenVMS binding	MCS$r_prop_basePartition
Defined By	CONTEXT
Required With **new**	Yes
Type	Relation
Data Type	MCS_ELEMENTID
Access	Write-once-at-creation
Relation Traversed	HAS_PARENT
Traversal Direction	To member
Inverse Property	None

baseType—Base Data Type

Indicates the data type upon which a data type is based.

Summary

Symbolic Name	
C binding	MCS_prop_baseType
OpenVMS binding	MCS$r_prop_baseType
Tag	NAD$K_ATT_SUBTYPE
Defined By	DATA_TYPE
Required With **new**	No
Type	Normal
Data Type	MCS_SMALLINT
Access	Write-once-at-creation

Constant Values

Value	Meaning
MCS_BASETYPE_SMALLINT	16-bit integer
MCS_BASETYPE_LONGINT	32-bit integer
MCS_BASETYPE_BOOLEAN	Boolean value (stored as 32-bit)
MCS_BASETYPE_FLOAT	F floating value
MCS_BASETYPE_DOUBLE	D floating value
MCS_BASETYPE_STRING	Null-terminated string
MCS_BASETYPE_STRINGDSC	String descriptor
MCS_BASETYPE_DATETIME	32-bit CS_DATE_TIME value
MCS_BASETYPE_MEMBLOCK	Pointer to memblock descriptor
MCS_BASETYPE_SCAN	Pointer to scan handle
MCS_BASETYPE_LIST	Pointer to list handle
MCS_BASETYPE_ARGSPEC	Argument descriptor
MCS_BASETYPE_ELEMENTID	Element ID
MCS_BASETYPE_NACTIME	64-bit NAC datetime value
MCS_BASETYPE_UNSPECIFIED	Unspecified data type

baseTypeSize—Base Data Type Size

Indicates the size of the base type in bytes.

Summary

Symbolic Name	
C binding	MCS_prop_baseTypeSize
OpenVMS binding	MCS$r_prop_baseTypeSize
Tag	NAD$K_ATT_LENGTH
Defined By	DATA_TYPE
Required With **new**	No
Type	Normal
Data Type	MCS_SMALLINT
Access	Write-once-at-creation

branchName—Branch Name

Contains a version's branch information, not including the version number. For example, for element **xyz(3:a:4:b:2)**, the value of **branchName** is "3:a:4:b".

Summary

Symbolic Name	
C binding	MCS_prop_branchName
OpenVMS binding	MCS$r_prop_branchName
Tag	NAD$K_ATT_BRANCH_NAME
Defined By	VERSION
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Read-only

checkout—Reserved Versions

Identifies the VERSION elements reserved by a context. The value of this property changes as versions are reserved and replaced.

Summary

Symbolic Name	
C binding	MCS_prop_checkout
OpenVMS binding	MCS$r_prop_checkout
Defined By	CONTEXT
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	RESERVED_BY
Traversal Direction	To owner
Inverse Property	**reservedBy**

childPartitions—Child Partitions

Identifies child partitions of a partition. The value of this property changes as the result of new partitions specifying this partition as parent.

childPartitions

Summary

Symbolic Name	
C binding	MCS_prop_childPartitions
OpenVMS binding	MCS$r_prop_childPartitions
Defined By	PARTITION
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	HAS_PARENT
Traversal Direction	To owner
Inverse Property	**parentPartition**

compPropDef—Computed Properties

Identifies all the computed properties belonging to an element type, including those inherited by the type. This value is a subset of the value of **propDef**.

Summary

Symbolic Name	
C binding	MCS_prop_compPropDef
OpenVMS binding	MCS$r_prop_compPropDef
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_COMPUTED_PROPERTY
Traversal Direction	To member
Inverse Property	**typesHavingCompProp**

contextDir—Context Directory

Identifies the context directory. Changing the value of this property with **setProp** also renames the context directory in the file system.

Summary

Symbolic Name	
C binding	MCS_prop_contextDir
OpenVMS binding	MCS$r_prop_contextDir
Tag	NAD$K_ATT_CONTEXTDIR
Defined By	CONTEXT
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Read/write

contextName—Context Name

Identifies the context that was active when an EVENT element was created.

Summary

Symbolic Name	
C binding	MCS_prop_contextName
OpenVMS binding	MCS$r_prop_contextName
Tag	NAD$K_ATT_HISTORY_CTXT_NAME
Defined By	EVENT
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Read/write

controlled—Version is Controlled

Indicates whether the version is controlled. Controlled versions cannot be changed in place; they must be reserved, changed, and replaced.

Summary

Symbolic Name	
C binding	MCS_prop_controlled
OpenVMS binding	MCS$r_prop_controlled
Tag	NAD$K_ATT_CONTROLLED
Defined By	VERSION
Required With **new**	No
Type	Normal
Data Type	MCS_BOOLEAN
Access	Read-only

Constant Values

Value	Meaning
TRUE	Version is controlled.
FALSE	Version is not controlled.

CPUTime—CPU Time Used

Indicates the amount of CPU time used by a tool invoked by a method in 10-millisecond ticks.

Summary

Symbolic Name	
C binding	MCS_prop_CPUTime
OpenVMS binding	MCS$r_prop_CPUTime
Tag	NAD$K_ATT_CPUTIME
Defined By	METHOD_INVOCATION
Required With **new**	No
Type	Normal
Data Type	MCS_LONGINT
Access	Write-once-at-creation

createdDate—Creation Date and Time

Indicates the date and time that an EVENT element was created.

Summary

Symbolic Name	
C binding	MCS_prop_createdDate
OpenVMS binding	MCS$r_prop_createdDate
Tag	NAD$K_ATT_CREATED_TIME
Defined By	EVENT METHOD_INVOCATION NAMED_ELEMENT
Required With **new**	No
Type	Normal
Data Type	MCS_DATETIME
Access	Write-once-at-creation

currCollection—Current Collection

Identifies the current open collection for a persistent process. Its value changes as the result of opening and closing collections.

Summary

Symbolic Name	
C binding	MCS_prop_currCollection
OpenVMS binding	MCS$r_prop_currCollection
Defined By	PERSISTENT_PROCESS
Required With **new**	No
Type	Relation
Data Type	MCS_ELEMENTID
Access	Read-only
Relation Traversed	HAS_CURR_COLLECTION
Traversal Direction	To member
Inverse Property	**ppForCollection**

currContext—Current Context

Identifies the context used by a persistent process.

Summary

Symbolic Name	
C binding	MCS_prop_currContext
OpenVMS binding	MCS$r_prop_currContext
Defined By	PERSISTENT_PROCESS
Required With **new**	No
Type	Relation
Data Type	MCS_ELEMENTID
Access	Read/write
Relation Traversed	HAS_CONTEXT
Traversal Direction	To member
Inverse Property	**ppForContext**

dataType—Data Type

For MSGARG elements, indicates the data type of the message argument.

For PROPERTY_TYPE elements, indicates the data type returned when the property is accessed using **getProp** or stored using **setProp**.

Summary

Symbolic Name	
C binding	MCS_prop_dataType
OpenVMS binding	MCS$r_prop_dataType
Defined By	MSGARG PROPERTY_TYPE
Required With **new**	Yes
Type	Relation
Data Type	MCS_ELEMENTID
Access	Write-once-at-creation
Relation Traversed	HAS_DATATYPE
Traversal Direction	To member
Inverse Property	**dataTypeUsers**

defaultAccess—Default Access

For contexts, identifies the default access control list to be applied to new elements created in this context if a default access control list was not supplied by the persistent process that opened this context.

For repositories (DATABASE), identifies the default access control list to be applied to new elements created in this repository if a default access control list was not supplied by the persistent process or context.

For persistent processes, identifies the default access control list to be applied to new objects created by this persistent process.

Note

See the description of the **access** property for information about the format of an access control list.

Summary

Symbolic Name	
C binding	MCS_prop_defaultAccess
OpenVMS binding	MCS$r_prop_defaultAccess
Tag	NAD$K_ATT_DEFAULT_ACL
Defined By	CONTEXT DATABASE PERSISTENT_PROCESS
Required With **new**	No
Type	Normal
Data Type	MCS_MEMBLOCK
Access	Read/write

defaultAttachment—Default Composite Attachment

Specifies a context's default attachment policy to be applied to VERSION elements if no attachment is specified in the relationship that connects a version to a composite.

Summary

Symbolic Name	
C binding	MCS_prop_defaultAttachment
OpenVMS binding	MCS$r_prop_defaultAttachment
Tag	NAD$K_ATT_DEFAULT_ATTACH
Defined By	CONTEXT
Required With **new**	No
Type	Normal
Data Type	MCS_SMALLINT
Access	Read/write

Constant Values

Value	Meaning
MCS_ATTACH_SPEC_VERSION	Attach specific version.
MCS_ATTACH_LAST_CHKIN	Attach last checked-in version.
MCS_ATTACH_LATEST	Attach reserved or last checked-in version.

The values of the **defaultAttachment** property are mutually exclusive.

definedLegalMembers—Defined Relation Members

Identifies the element types that are defined to be members of a relation. The scan does not include subtypes of the defined types. (The value of the **legalMembers** property does include subtypes.)

definedLegalMembers

Summary

Symbolic Name	
C binding	MCS_prop_definedLegalMembers
OpenVMS binding	MCS$r_prop_definedLegalMembers
Defined By	RELATION_TYPE
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

---

definedLegalOwners—Defined Relation Owners

Identifies the element types that are defined to be owners of a relation. The scan does not include subtypes of the defined types. (The value of the **legalOwners** property does include subtypes.)

Summary

Symbolic Name	
C binding	MCS_prop_definedLegalOwners
OpenVMS binding	MCS$r_prop_definedLegalOwners
Defined By	RELATION_TYPE
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

definedMethods—Methods

Identifies methods that implement messages for an element type. The scan does not include inherited methods. (The **methods** property does include inherited methods.)

Summary

Symbolic Name	
C binding	MCS_prop_definedMethods
OpenVMS binding	MCS$r_prop_definedMethods
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

definedPropDefs—Defined Properties

Identifies all properties defined by an element type. The scan does not include property definitions inherited from supertypes. (The **propDef** property includes inherited properties.)

Summary

Symbolic Name	
C binding	MCS_prop_definedPropDefs
OpenVMS binding	MCS$r_prop_definedPropDefs
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

deltaFile—Delta File

Gives the name and location of the delta file that contains the file version represented by a BINARY element. The method that implements **new** for BINARY elements sets this property if the file is stored internally, and leaves the property null if the file is stored externally. Users cannot supply a value for this property.

Summary

Symbolic Name	
C binding	MCS_prop_deltaFile
OpenVMS binding	MCS$r_prop_deltaFile
Tag	NAD$K_ATT_DELTAFILE
Defined By	BINARY
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Write-once-at-creation

dependencies—Dependencies

Identifies the elements that immediately depend on the element possessing the property.

Summary

Symbolic Name	
C binding	MCS_prop_dependencies
OpenVMS binding	MCS$r_prop_dependencies
Defined By	VERSION
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	DEPENDS_ON
Traversal Direction	To owner
Inverse Property	**dependents**

dependents—Dependents

Identifies the elements on which the element possessing the property immediately depends.

Summary

Symbolic Name	
C binding	MCS_prop_dependents
OpenVMS binding	MCS$r_prop_dependents
Defined By	VERSION
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	DEPENDS_ON
Traversal Direction	To member
Inverse Property	**dependencies**

derivedFrom—Derived From

For a VERSION or subtype element, identifies the METHOD_INVOCATION element that recorded the creation of this derived object.

For a METHOD_INVOCATION element, identifies the source VERSION elements that this method invocation processed.

Summary

Symbolic Name	
C binding	MCS_prop_derivedFrom
OpenVMS binding	MCS$r_prop_derivedFrom
Defined By	METHOD_INVOCATION VERSION
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Write-once-at-creation
Relation Traversed	METHOD_INPUT (when on METHOD_INVOCATION) METHOD_OUTPUT (when on VERSION)
Traversal Direction	To member
Inverse Property	**derives**

derives—Derives

For a VERSION or subtype element, identifies the METHOD_INVOCATION elements that have processed this source object.

For a METHOD_INVOCATION element, identifies the VERSION elements that resulted from this method invocation.

Summary

Symbolic Name	
C binding	MCS_prop_derives
OpenVMS binding	MCS$r_prop_derives
Defined By	METHOD_INVOCATION VERSION
Required With **new**	Yes (on METHOD_INVOCATION) No (on VERSION)
Type	Relation
Data Type	MCS_SCAN
Access	Write-once-at-creation
Relation Traversed	METHOD_OUTPUT (when on METHOD_INVOCATION) METHOD_INPUT (when on VERSION)
Traversal Direction	To owner
Inverse Property	**derivedFrom**

description—Element Description

Lists strings that describe the element. This property is available for users who want to provide a general description for an element.

Summary

Symbolic Name	
C binding	MCS_prop_description
OpenVMS binding	MCS$r_prop_description
Tag	NAD$K_ATT_DESCRIPTION
Defined By	NAMED_ELEMENT
Required With **new**	No
Type	Normal
Data Type	MCS_LIST
Access	Read/write

direction—Relation Traversal Direction

Specifies the direction of relation traversal. For a relation property, the traversal will go either from owner to member or from member to owner. For closure properties, the traversal goes to all members or all owners.

Summary

Symbolic Name	
C binding	MCS_prop_direction
OpenVMS binding	MCS$r_prop_direction
Tag	NAD$K_ATT_DIRECTION
Defined By	HAS_RELATION_PROPERTY
Required With **new**	Yes
Type	Normal
Data Type	MCS_SMALLINT
Access	Read/write

Constant Values

Value	Meaning
MCS_DIRECTION_TO_MEMBER	Traverse the relation from owner to member.
MCS_DIRECTION_TO_ALL_MEMBERS	Traverse the relation from owner to all members.
MCS_DIRECTION_TO_OWNER	Traverse the relation from member to owner.
MCS_DIRECTION_TO_ALL_OWNERS	Traverse the relation from member to all owners.

elapsedTime—Elapsed Time

Indicates the total elapsed time for processing associated with a method invocation in units of 10-millisecond ticks.

Summary

Symbolic Name	
C binding	MCS_prop_elapsedTime
OpenVMS binding	MCS$r_prop_elapsedTime
Tag	NAD$K_ATT_ELAPSEDTIME
Defined By	METHOD_INVOCATION
Required With **new**	No
Type	Normal
Data Type	MCS_LONGINT
Access	Write-once-at-creation

elementType—Element Type

Identifies the ELEMENT_TYPE element that represents an element's type. The value of this property is established when the element is created.

Summary

Symbolic Name	
C binding	MCS_prop_elementType
OpenVMS binding	MCS$r_prop_elementType
Defined By	ELEMENT
Required With **new**	No
Type	Computed
Data Type	MCS_ELEMENTID
Access	Read-only

filePath—File Path

Contains the native file system specification of the file represented by a BINARY element. Oracle CDD/Repository computes the value of this property in one of the following ways:

- If the file is stored externally (**storeType** is MCS_STORETYPE_ EXTERNAL), the value of the **filePath** property is equal to the value of the **storedIn** property.
- If the file is stored internally (**storeType** is MCS_STORETYPE_ INTERNAL), the value of the property is computed in the following manner:
  - When a file is opened, a read-only copy is created. Also, a file system link to this file is created from the directory that corresponds to the Oracle CDD/Repository directory that contains the element. The value of the **filePath** property is the file specification of this file system link.
  - When a file is reserved, a read/write copy is created in the directory that corresponds to the Oracle CDD/Repository directory that contains the element. The value of the **filePath** property is the file specification of this file.

Summary

Symbolic Name	
C binding	MCS_prop_filePath
OpenVMS binding	MCS$r_prop_filePath
Defined By	BINARY
Required With **new**	No
Type	Computed
Data Type	MCS_LIST
Access	Read-only

firstVersion—First Version on This Branch

Identifies the first version on the same branch as the version possessing the property.

Summary

Symbolic Name	
C binding	MCS_prop_firstVersion
OpenVMS binding	MCS$r_prop_firstVersion
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_ELEMENTID
Access	Read-only

freezeTime—Replacement Time

Indicates the time at which a VERSION element was replaced.

Summary

Symbolic Name	
C binding	MCS_prop_freezeTime
OpenVMS binding	MCS$r_prop_freezeTime
Tag	NAD$K_ATT_FREEZE_TIME
Defined By	VERSION
Required With **new**	No
Type	Normal
Data Type	MCS_DATETIME
Access	Read-only

funcType—Method Function Type

Identifies a method's function type. The function type determines how a method carries out its processing.

Summary

Symbolic Name	
C binding	MCS_prop_funcType
OpenVMS binding	MCS$r_prop_funcType
Tag	NAD$K_ATT_FUNCTYPE
Defined By	METHOD
Required With **new**	Yes
Type	Normal
Data Type	MCS_SMALLINT
Access	Read/write

Constant Values

Value	Meaning
MCS_METHOD_INTERNAL_CODE	Invokes a subroutine. The subroutine is called directly by the message dispatch mechanism.
MCS_METHOD_EXTERNAL_CODE	Invokes a subroutine that the method dispatch mechanism loads into or otherwise associates with the executing program's address space before calling it.
MCS_METHOD_EXTERNAL_PROGRAM	Invokes an external program (image or command script) as a separate process through a mechanism appropriate for the host platform.

Value	Meaning
MCS_METHOD_INTERNAL_SCRIPT	Invokes an internal script. This method type is not supported in this release of Oracle CDD/Repository.
MCS_METHOD_NULL	Invokes a null function.
MCS_METHOD_TRANSPARENT	Invokes the method that would be invoked if no method were defined for this message–type pair. Transparent methods serve as placeholders for preambles and postambles. This method type is not supported in this release of Oracle CDD/Repository.
MCS_METHOD_SUPEROP	Invokes the supertype's method for this message. This function type provides a convenient means of creating a placeholder for preambles and postambles.
MCS_METHOD_ILLEGAL	Specifies an illegal function. Invoking the method results in an error indicating that no method was specified for the message.

hasChildren—Composite Children

Identifies the versions that are members of a composite. The value of this property changes as the result of **attach** and **detach** messages.

Summary

Symbolic Name	
C binding	MCS_prop_hasChildren
OpenVMS binding	MCS$r_prop_hasChildren
Defined By	COMPOSITE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	COMPOSITE_PART
Traversal Direction	To member
Inverse Property	**hasParents**

hasParents—Parent Composites

Identifies the composites to which a version is attached. The value of this property changes as the result of **attach** and **detach** messages.

Summary

Symbolic Name	
C binding	MCS_prop_hasParents
OpenVMS binding	MCS$r_prop_hasParents
Defined By	VERSION
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	COMPOSITE_PART
Traversal Direction	To owner
Inverse Property	**hasChildren**

history—History

Identifies all the EVENT elements for an element.

Summary

Symbolic Name	
C binding	MCS_prop_history
OpenVMS binding	MCS$r_prop_history
Defined By	NAMED_ELEMENT
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	HISTORY_LIST
Traversal Direction	To member
Inverse Property	**historyOwner**

historyComment—History Comment

Contains the comment (in the form of a list of strings) supplied by the user.

Summary

Symbolic Name	
C binding	MCS_prop_historyComment
OpenVMS binding	MCS$r_prop_historyComment
Tag	NAD$K_ATT_HISTORY_TEXT
Defined By	EVENT
Required With **new**	No
Type	Normal
Data Type	MCS_LIST
Access	Write-once-at-creation

implementsMessage—Message Implemented by Method

Identifies the message for which this method is the implementation.

Summary

Symbolic Name	
C binding	MCS_prop_implementsMessage
OpenVMS binding	MCS$r_prop_implementsMessage
Defined By	METHOD
Required With **new**	No
Type	Relation
Data Type	MCS_ELEMENTID
Access	Read/write
Relation Traversed	HAS_MESSAGE
Traversal Direction	To member
Inverse Property	**implementingMethods**

implementsMethod—Method Implementing Computed Property

Identifies the method that computes the value of a computed property.

Summary

Symbolic Name	
C binding	MCS_prop_implementsMethod
OpenVMS binding	MCS$r_prop_implementsMethod
Defined By	HAS_COMPUTED_PROPERTY
Required With **new**	Yes
Type	Relation
Data Type	MCS_ELEMENTID
Access	Read/write
Relation Traversed	IMPLEMENTS_METHOD
Traversal Direction	To member
Inverse Property	**compPropUsing**

implementsRelation—Relation Implementing Relation Property

Identifies the RELATION_TYPE element that implements a relation property.

Summary

Symbolic Name	
C binding	MCS_prop_implementsRelation
OpenVMS binding	MCS$r_prop_implementsRelation
Defined By	HAS_RELATION_PROPERTY
Required With **new**	Yes
Type	Relation
Data Type	MCS_ELEMENTID
Access	Read/write
Relation Traversed	IMPLEMENTS_RELATION
Traversal Direction	To member
Inverse Property	**relPropUsing**

importedFrom—Imported from File

Contains the file specification of the file in the native file system from which the element was last imported. The **import** message sets this property.

Summary

Symbolic Name	
C binding	MCS_prop_importedFrom
OpenVMS binding	MCS$r_prop_importedFrom
Tag	NAD$K_ATT_IMPORTEDFROM
Defined By	BINARY
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Write-once-at-creation

inPartition—Partition

Identifies the partition in which a version resides. The value of this property is set to the current base partition when a new version is replaced, and changes thereafter as the result of **promote** messages.

Summary

Symbolic Name	
C binding	MCS_prop_inPartition
OpenVMS binding	MCS$r_prop_inPartition
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_ELEMENTID
Access	Read-only

instances—Instances of Type/Instances in Partition

For ELEMENT_TYPE and its subtypes, identifies all instances of the type. (Compare with **allInstances**, which includes instances of subtypes.)

For PARTITION, identifies the versions that reside in a partition.

Summary

Symbolic Name	
C binding	MCS_prop_instances
OpenVMS binding	MCS$r_prop_instances
Defined By	ELEMENT_TYPE PARTITION
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

invocationStatus—Method Invocation Status

Contains the status code returned by the tool invoked by a method, in cases where it is meaningful.

Summary

Symbolic Name	
C binding	MCS_prop_invocationStatus
OpenVMS binding	MCS$r_prop_invocationStatus
Tag	NAD$K_ATT_INVOCATIONSTATUS
Defined By	METHOD_INVOCATION
Required With **new**	No
Type	Normal
Data Type	MCS_LONGINT
Access	Write-once-at-creation

invocationString—Invocation String

For METHOD_INVOCATION elements, contains the actual invocation string that was sent to the tool.

For TOOL elements, specifies a template for a command line to be executed for external program methods, or a routine name for external or internal code methods. For external program methods, the invocation string includes placeholders for parameters that Oracle CDD/Repository fills in with values from the element to which a message is sent. See the *Oracle CDD/Repository Callable Interface Manual* for the format of invocation strings.

Summary

Symbolic Name	
C binding	MCS_prop_invocationString
OpenVMS binding	MCS$r_prop_invocationString
Tag	NAD$K_ATT_INVOCAT_STR
Defined By	METHOD_INVOCATION TOOL
Required With **new**	Yes (must be non-NULL for methods of type EXTERNAL_CODE, INTERNAL_CODE, or EXTERNAL_PROGRAM)
Type	Normal
Data Type	MCS_STRING
Access	Read/write

invokes—Invokes Tool

Identifies the TEXT_TOOL or BINARY_TOOL element that represents the physical implementation of a tool.

Summary

Symbolic Name	
C binding	MCS_prop_invokes
OpenVMS binding	MCS$r_prop_invokes
Defined By	TOOL
Required With **new**	No
Type	Relation
Data Type	MCS_ELEMENTID
Access	Read/write
Relation Traversed	INVOKES_TOOL
Traversal Direction	To member
Inverse Property	**invokedBy**

keepHist—Keep History

Indicates if an EVENT element is to be created each time the method possessing the property is executed. If no value is specified for this property, no history is kept.

Summary

Symbolic Name	
C binding	MCS_prop_keepHist
OpenVMS binding	MCS$r_prop_keepHist
Tag	NAD$K_ATT_KEEPHIST
Defined By	METHOD
Required With **new**	No
Type	Normal
Data Type	MCS_BOOLEAN
Access	Read/write

keepHist

Constant Values

Value	Meaning
TRUE	A history record will be created each time the method is executed.
FALSE	A history record will not be created each time the method is executed.

lastVersion—Last Version

Identifies the most recent visible version element on a line of descent, regardless of replacement status. (Recall that the contents of a reserved version are visible only to the person who reserved it.) The value of this property is the same for any version on a given line of descent.

Summary

Symbolic Name	
C binding	MCS_prop_lastVersion
OpenVMS binding	MCS$r_prop_lastVersion
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_ELEMENTID
Access	Read-only

legalMembers—Legal Relation Members

Identifies all the element types that can be members of a relation, including subtypes of the types that are defined as members.

Summary

Symbolic Name	
C binding	MCS_prop_legalMembers
OpenVMS binding	MCS$r_prop_legalMembers
Defined By	RELATION_TYPE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	RELATION_MEMBER
Traversal Direction	To member
Inverse Property	**relationMember**

legalOwners—Legal Relation Owners

Identifies all the element types that can be owners of a relation, including subtypes of the types that are defined as owners.

legalOwners

Summary

Symbolic Name	
C binding	MCS_prop_legalOwners
OpenVMS binding	MCS$r_prop_legalOwners
Defined By	RELATION_TYPE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_RELATION
Traversal Direction	To owner
Inverse Property	**ownsRelation**

---

logFile—Log File

Names a file that contains the execution log of the tool invoked in this method invocation.

Summary

Symbolic Name	
C binding	MCS_prop_logFile
OpenVMS binding	MCS$r_prop_logFile
Defined By	METHOD_INVOCATION
Required With **new**	No
Type	Relation
Data Type	MCS_STRING
Access	Write-once-at-creation

messageName—Message Name

Names the message that caused an EVENT element to be created.

Summary

Symbolic Name	
C binding	MCS_prop_messageName
OpenVMS binding	MCS$r_prop_messageName
Tag	NAD$K_ATT_HISTORY_MSG_NAME
Defined By	EVENT
Required With **new**	Yes
Type	Normal
Data Type	MCS_STRING
Access	Read/write

methods—Methods

Identifies all methods that operate on the type, including those inherited without refinement.

Summary

Symbolic Name	
C binding	MCS_prop_methods
OpenVMS binding	MCS$r_prop_methods
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_DEFAULT_METHOD
Traversal Direction	To member
Inverse Property	**typesUsingMethod**

methodType—Method Type

Identifies the element type for which a method implements a message.

Summary

Symbolic Name	
C binding	MCS_prop_methodType
OpenVMS binding	MCS$r_prop_methodType
Defined By	METHOD
Required With **new**	No
Type	Computed
Data Type	MCS_ELEMENTID
Access	Read-only

methodUsed—Method Used

Identifies the METHOD element that was invoked in this method invocation.

Summary

Symbolic Name	
C binding	MCS_prop_methodUsed
OpenVMS binding	MCS$r_prop_methodUsed
Defined By	ATIS_METHOD_INVOC
Required With **new**	No
Type	Relation
Data Type	MCS_ELEMENTID
Access	Write-once-at-creation
Relation Traversed	HAS_METHOD_USED
Traversal Direction	To member
Inverse Property	**miUsingMethod**

msgSent—Message Sent

Identifies the MESSAGE element that was sent to start this method invocation.

Summary

Symbolic Name	
C binding	MCS_prop_msgSent
OpenVMS binding	MCS$r_prop_msgSent
Defined By	ATIS_METHOD_INVOC
Required With **new**	No
Type	Relation
Data Type	MCS_ELEMENTID
Access	Write-once-at-creation
Relation Traversed	HAS_MSG_SENT
Traversal Direction	To member
Inverse Property	**miForMessage**

msgTarget—Message Target

Identifies the VERSION (or subtype) element that was the target of the message that started this method invocation.

msgTarget

Summary

Symbolic Name	
C binding	MCS_prop_msgTarget
OpenVMS binding	MCS$r_prop_msgTarget
Defined By	ATIS_METHOD_INVOC
Required With **new**	No
Type	Relation
Data Type	MCS_ELEMENTID
Access	Write-once-at-creation
Relation Traversed	HAS_MSG_TARGET
Traversal Direction	To member
Inverse Property	**miForTarget**

mutable—Property is Mutable

Specifies whether a property belonging to a VERSION element can be changed even if the element is immutable (replaced).

Summary

Symbolic Name	
C binding	MCS_prop_mutable
OpenVMS binding	MCS$r_prop_mutable
Tag	NAD$K_ATT_MUTABLE
Defined By	HAS_PROPERTY
Required With **new**	No
Type	Normal
Data Type	MCS_BOOLEAN
Access	Read/write

Constant Values

Value	Meaning
TRUE	The property can be changed even if the owning element is replaced.
FALSE	The property cannot be changed when the owning element is replaced.

name—Element Name

Contains the instance name of the element. The value of this property is established when the element is created, and can be changed either by using **setProp** or by sending the **rename** message. The value of the **name** property is the same as the first entry in the value of the **alternateNames** property.

Summary

Symbolic Name	
C binding	MCS_prop_name
OpenVMS binding	MCS$r_prop_name
Tag	NAD$K_ATT_NAD_MCSNAME
Defined By	NAMED_ELEMENT
Required With **new**	Yes, for the following subtypes of NAMED_ELEMENT: BINARY, COLLECTION, CONTEXT, DIRECTORY, MESSAGE, METHOD, MSGARG, PARTITION, PERSISTENT_PROCESS, and TYPE
Type	Normal
Data Type	MCS_STRING
Access	Read/write

nextVersions—Next Versions

Identifies the logical descendant versions of this version. Logical descendants include the next version on the line of descent, any versions that branch from this version, and any versions to which this version merges. If the next version on the line of descent exists, it is guaranteed to be the first in the scan.

The value of this property changes as the result of messages that manipulate versions in a component, such as **reserve** or **merge**.

Summary

Symbolic Name	
C binding	MCS_prop_nextVersions
OpenVMS binding	MCS$r_prop_nextVersions
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

node—Node

Identifies the network node on which the repository represented by this DATABASE element resides. The value is either the node name or the name service handle for the node, whichever is appropriate.

Summary

Symbolic Name	
C binding	MCS_prop_node
OpenVMS binding	MCS$r_prop_node
Tag	NAD$K_ATT_NODE_NAME
Defined By	DATABASE
Required With **new**	Yes
Type	Normal
Data Type	MCS_STRING
Access	Read/write

numChildren—Number of Composite Children

Indicates the number of versions attached to a composite.

Summary

Symbolic Name	
C binding	MCS_prop_numChildren
OpenVMS binding	MCS$r_prop_numChildren
Defined By	COMPOSITE
Required With **new**	No
Type	Computed
Data Type	MCS_LONGINT
Access	Read-only

openedBy—Opening Contexts

Identifies CONTEXT elements that have opened an element. The value of this property changes as contexts open and close the element.

Summary

Symbolic Name	
C binding	MCS_prop_openedBy
OpenVMS binding	MCS$r_prop_openedBy
Defined By	BINARY
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	OPENED_BY
Traversal Direction	To owner
Inverse Property	**openedFiles**

openedFiles—Opened Files

Identifies BINARY (or subtype) elements that a context has opened. The value of this property changes as the context's user opens and closes elements.

Summary

Symbolic Name	
C binding	MCS_prop_openedFiles
OpenVMS binding	MCS$r_prop_openedFiles
Defined By	CONTEXT
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	OPENED_BY
Traversal Direction	To member
Inverse Property	**openedBy**

optionsString—Options String

Indicates the values of all options that were used by the tool invoked during this method invocation, including settings obtained from default values.

Summary

Symbolic Name	
C binding	MCS_prop_optionsString
OpenVMS binding	MCS$r_prop_optionsString
Tag	NAD$K_ATT_OPTIONSSTRING
Defined By	METHOD_INVOCATION
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Write-once-at-creation

OSVersion—Operating System Version

Indicates the operating system version number under which this method invocation occurred.

Summary

Symbolic Name	
C binding	MCS_prop_OSVersion
OpenVMS binding	MCS$r_prop_OSVersion
Tag	NAD$K_ATT_OSVERSION
Defined By	METHOD_INVOCATION
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Write-once-at-creation

owner—Element Owner

Contains a system-specific identifier for the user who created the element.

Summary

Symbolic Name	
C binding	MCS_prop_owner
OpenVMS binding	MCS$r_prop_owner
Defined By	NAMED_ELEMENT
Required With **new**	No
Type	Computed
Data Type	System-specific identifier
Access	Read-only

ownsRelation—Relation Types Owned

Identifies all the relation types that can be owned by an element type, including subtypes of the relation types that are defined as owned.

Summary

Symbolic Name	
C binding	MCS_prop_ownsRelation
OpenVMS binding	MCS$r_prop_ownsRelation
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_RELATION
Traversal Direction	To member
Inverse Property	**legalOwners**

parentInContext—Parent Composite in Context

Identifies the COMPOSITE element(s) that immediately contain a version in the current context. There may be more than one element in the scan, since more than one composite can contain a version in a context.

Summary

Symbolic Name	
C binding	MCS_prop_parentInContext
OpenVMS binding	MCS$r_prop_parentInContext
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

parentPartition—Parent Partition

Identifies the parent partition of a partition. If the partition is the root of a partition hierarchy, the value returned is null.

Summary

Symbolic Name	
C binding	MCS_prop_parentPartition
OpenVMS binding	MCS$r_prop_parentPartition
Defined By	PARTITION
Required With **new**	No
Type	Relation
Data Type	MCS_ELEMENTID
Access	Write-once-at-creation
Relation Traversed	HAS_PARENT
Traversal Direction	To member
Inverse Property	**childPartitions**

partitionDir—Partition Directory

Names a partition's partition directory. Changing the value of this property with **setProp** also renames the partition directory in the native file system.

Summary

Symbolic Name	
C binding	MCS_prop_partitionDir
OpenVMS binding	MCS$r_prop_partitionDir
Tag	NAD$K_ATT_PARTITIONDIR
Defined By	PARTITION
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Read/write

passingMechanism—Argument Use

Indicates whether the argument is input, output, or both input and output.

Summary

Symbolic Name	
C binding	MCS_prop_passingMechanism
OpenVMS binding	MCS$r_prop_passingMechanism
Tag	NAD$K_ATT_MECHANISM
Defined By	HAS_MSGARG
Required With **new**	Yes
Type	Normal
Data Type	MCS_SMALLINT
Access	Write-once-at-creation

Constant Values

Value	Meaning
MCS_MSGARG_IN	Input argument, read-only.
MCS_MSGARG_OUT	Output argument, write-only.
MCS_MSGARG_INOUT	Input and output argument, read/write.

path—Repository Path

Contains the native file system path name of the repository, or its name service handle if a name service is available.

Summary

Symbolic Name	
C binding	MCS_prop_path
OpenVMS binding	MCS$r_prop_path
Tag	NAD$K_ATT_ANCHOR_NAME
Defined By	DATABASE
Required With **new**	Yes
Type	Normal
Data Type	MCS_STRING
Access	Read/write

pattern—Filespec Matching Pattern

Gives a pattern for matching file names, to determine automatically if files are of a particular type. The format of the pattern is that of a UNIX-style regular expression.

Summary

Symbolic Name	
C binding	MCS_prop_pattern
OpenVMS binding	MCS$r_prop_pattern
Tag	NAD$K_ATT_PATTERN
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Read/write

postamble—Method Postamble

Identifies methods to execute after the method possessing the property has executed. The methods identified by **postamble** may execute in any order.

Summary

Symbolic Name	
C binding	MCS_prop_postamble
OpenVMS binding	MCS$r_prop_postamble
Defined By	METHOD
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_POSTAMBLE
Traversal Direction	To member
Inverse Property	**methodUsingPostamble**

preamble—Method Preamble

Identifies methods to execute before the method possessing the property has executed. The methods identified by **preamble** may execute in any order.

preamble

Summary

Symbolic Name	
C binding	MCS_prop_preamble
OpenVMS binding	MCS$r_prop_preamble
Defined By	METHOD
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_PREAMBLE
Traversal Direction	To member
Inverse Property	**methodUsingPreamble**

prevVersions—Previous Versions

Identifies the versions that are logical ancestors to a version. Logical ancestors include the previous version on the line of descent, any version from which this version branches, and any versions that merge to this version. If the previous version on the line of descent exists, it is guaranteed to be the first in the scan.

The value of this property is established at the time a version is created and can change if another version merges to this one.

Summary

Symbolic Name	
C binding	MCS_prop_prevVersions
OpenVMS binding	MCS$r_prop_prevVersions
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

processingName—Processing Name

Contains the element name visible to an automatic processor when processing the element; for example, the field name seen by a language compiler on a field element, or the file name of a file element. May be overridden by other properties such as **filePath**.

Summary

Symbolic Name	
C binding	MCS_prop_processingName
OpenVMS binding	MCS$r_prop_processingName
Tag	NAD$K_ATT_PROCESSING_NAME
Defined By	ELEMENT
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Read/write

propDef—Properties

Identifies all properties defined for the type and for its supertypes.

propDef

Summary

Symbolic Name	
C binding	MCS_prop_propDef
OpenVMS binding	MCS$r_prop_propDef
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_PROPERTY
Traversal Direction	To member
Inverse Property	**typesHavingProp**

referenceCount—File Open Count

Indicates the number of contexts that have opened an element by sending it the **open** message.

Summary

Symbolic Name	
C binding	MCS_prop_referenceCount
OpenVMS binding	MCS$r_prop_referenceCount
Defined By	BINARY
Required With **new**	No
Type	Computed
Data Type	MCS_LONGINT
Access	Read-only

related—Related Partitions

Identifies alternate parent partitions of a partition.

Summary

Symbolic Name	
C binding	MCS_prop_related
OpenVMS binding	MCS$r_prop_related
Defined By	PARTITION
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_RELATED_PARTITION
Traversal Direction	To member
Inverse Property	**partitionsRelatedTo**

relationMember—Relation Types Member Of

Identifies all the relation types (including subtypes) of which an element type can be a member.

relationMember

Summary

Symbolic Name	
C binding	MCS_prop_relationMember
OpenVMS binding	MCS$r_prop_relationMember
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	RELATION_MEMBER
Traversal Direction	To owner
Inverse Property	**legalMembers**

relMember—Relationship Members

Identifies the members of a relationship. For dependency relationships (DEPENDS_ON and subtypes), when a member participant of a relationship changes, the relationship owner(s) must be notified. This is a normal property, but its data type is MCS_SCAN.

Summary

Symbolic Name	
C binding	MCS_prop_relMember
OpenVMS binding	MCS$r_prop_relMember
Tag	NAD$K_ATT_REL_MEMBER
Defined By	RELATION
Required With **new**	Yes
Type	Normal
Data Type	MCS_SCAN
Access	Read/write

relOwner—Relationship Owners

Identifies the owners of a relationship. For dependency relationships (DEPENDS_ON and subtypes), owners are the elements that are notified when any of the member participants change. This is a normal property, but its data type is MCS_SCAN.

Summary

Symbolic Name	
C binding	MCS_prop_relOwner
OpenVMS binding	MCS$r_prop_relOwner
Tag	NAD$K_ATT_REL_OWNER
Defined By	RELATION
Required With **new**	Yes
Type	Normal
Data Type	MCS_SCAN
Access	Read/write

relPropDef—Relation Properties

Identifies all the relation and closure properties belonging to an element type, including those inherited by the type. This value is a subset of the value of **propDef**.

relPropDef

Summary

Symbolic Name	
C binding	MCS_prop_relPropDef
OpenVMS binding	MCS$r_prop_relPropDef
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read/write
Relation Traversed	HAS_RELATION_PROPERTY
Traversal Direction	To member
Inverse Property	**typesHavingRelProp**

required—Required Value

Indicates whether a property value must be supplied when an element is created, or whether an argument is required with a message.

Summary

Symbolic Name	
C binding	MCS_prop_required
OpenVMS binding	MCS$r_prop_required
Tag	NAD$K_ATT_REQUIRED
Defined By	HAS_MSGARG HAS_PROPERTY
Required With **new**	Yes
Type	Normal
Data Type	MCS_SMALLINT
Access	Read/write

required

Constant Values

Value	Meaning
TRUE	A value is required.
FALSE	A value is not required.

rootBranchInstances—Root Branch Instances

Identifies all root branches for instances of the type (VERSION subtypes only). The root branch of a component is the first version on the component's main line of descent.

Summary

Symbolic Name	
C binding	MCS_prop_rootBranchInstances
OpenVMS binding	MCS$r_prop_rootBranchInstances
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Computed
Data Type	MCS_SCAN
Access	Read-only

rootBranchName—Root Branch Name

Contains the name of a VERSION (or subtype) element, stripped of version, branch, and directory information.

rootBranchName

Summary

Symbolic Name	
C binding	MCS_prop_rootBranchName
OpenVMS binding	MCS$r_prop_rootBranchName
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_STRING
Access	Read-only

---

rootPath—File System Root Directory

Names the file system directory that is the root of the file system associated with a repository.

Summary

Symbolic Name	
C binding	MCS_prop_rootPath
OpenVMS binding	MCS$r_prop_rootPath
Tag	NAD$K_ATT_ANCHORROOT_NAME
Defined By	DATABASE
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Write-once

rootVersion—First Version on Main Line of Descent

Identifies the first version on the main line of descent of the component containing the version possessing the property.

Summary

Symbolic Name	
C binding	MCS_prop_rootVersion
OpenVMS binding	MCS$r_prop_rootVersion
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_ELEMENTID
Access	Read-only

scale—Scale Factor

Contains the scale factor for numeric property values. If supplied, the value of the property is multiplied by the scale.

Summary

Symbolic Name	
C binding	MCS_prop_scale
OpenVMS binding	MCS$r_prop_scale
Tag	NAD$K_ATT_SCALE
Defined By	PROPERTY_TYPE
Required With **new**	No
Type	Normal
Data Type	MCS_SMALLINT
Access	Read/write

scalingFactor—CPU Power Scaling Factor

Contains a figure of merit indicating the relative processing power of a CPU. It is used in build scheduling.

Summary

Symbolic Name	
C binding	MCS_prop_scalingFactor
OpenVMS binding	MCS$r_prop_scalingFactor
Tag	NAD$K_ATT_SCALINGFACTOR
Defined By	METHOD_INVOCATION
Required With **new**	No
Type	Normal
Data Type	MCS_FLOAT
Access	Write-once-at-creation

simpleName—Simple Name

Contains the name of an element without any repository or directory information at the front.

Summary

Symbolic Name	
C binding	MCS_prop_simpleName
OpenVMS binding	MCS$r_prop_simpleName
Defined By	NAMED_ELEMENT
Required With **new**	No
Type	Computed
Data Type	MCS_STRING
Access	Read-only

status—Reservation Status

Indicates the reservation status of a version. The value changes as the result of **reserve**, **replace**, and **unreserve** messages.

Summary

Symbolic Name	
C binding	MCS_prop_status
OpenVMS binding	MCS$r_prop_status
Tag	NAD$K_ATT_STATUS
Defined By	VERSION
Required With **new**	No
Type	Normal
Data Type	MCS_LONGINT
Access	Read-only

Constant Values

Value	Meaning
MCS_STS_AVAIL	Available for reserve—The version is last version on the line of descent and is replaced.
MCS_STS_RO	Read-only—The version has descendants on the same branch, and may be reserved only if a new branch is created.
MCS_STS_GHOST	The version is reserved.
MCS_STS_FROZEN	The version may not be reserved.

The values of the **status** property are mutually exclusive.

storedIn—File System Location

Contains the native file system location of an externally stored file. Changing the value of this property with **setProp** does *not* change the file system location of the file. The user is responsible for ensuring that the property value agrees with the actual file location.

For internally stored files, this property is meaningless.

Summary

Symbolic Name	
C binding	MCS_prop_storedIn
OpenVMS binding	MCS$r_prop_storedIn
Tag	NAD$K_ATT_STOREDIN
Defined By	BINARY
Required With **new**	Yes (if the value of **storeType** is MCS_STORETYPE_EXTERNAL)
Type	Normal
Data Type	MCS_STRING
Access	Read/write

storeType—File Storage Type

Indicates how the file represented is stored: externally, or internally under Oracle CDD/Repository control.

If the file is stored externally, Oracle CDD/Repository contains only the file specification of the file as the value of the **storedIn** property. No attempt is made to control access to the file.

If the file is stored internally, Oracle CDD/Repository maintains the file in delta-file format; the file may be accessed from directories controlled by Oracle CDD/Repository. See the descriptions of the **filePath** property, and the **open**, **reserve**, **promote**, **close**, **replace**, and **unreserve** messages for information about this behavior.

Summary

Symbolic Name	
C binding	MCS_prop_storeType
OpenVMS binding	MCS$r_prop_storeType
Tag	NAD$K_ATT_STORETYPE
Defined By	BINARY
Required With **new**	Yes
Type	Normal
Data Type	MCS_SMALLINT
Access	Write-once-at-creation

Constant Values

Value	Meaning
MCS_STORETYPE_EXTERNAL	File is under Oracle CDD/Repository control.
MCS_STORETYPE_INTERNAL	File is not under Oracle CDD/Repository control.

subTypes—Type Subtypes

Identifies all the ELEMENT_TYPE elements representing types that are specializations of a type. This value changes as new types are added that specify this type as supertype.

subTypes

Summary

Symbolic Name	
C binding	MCS_prop_subTypes
OpenVMS binding	MCS$r_prop_subTypes
Defined By	TYPE
Required With **new**	No
Type	Relation
Data Type	MCS_SCAN
Access	Read-only
Relation Traversed	HAS_SUPER_TYPE
Traversal Direction	To owner
Inverse Property	**superTypes**

superTypes—Type Supertypes

Identifies the ELEMENT_TYPE element specified as supertype when an element type was created.

Note

This property is a scan to allow for possible future extension to multiple inheritance. The property is currently restricted to having a single value.

Summary

Symbolic Name	
C binding	MCS_prop_superTypes
OpenVMS binding	MCS$r_prop_superTypes
Defined By	TYPE
Required With **new**	Yes (required by ELEMENT_TYPE)
Type	Relation
Data Type	MCS_SCAN
Access	Write-once-at-creation
Relation Traversed	HAS_SUPER_TYPE
Traversal Direction	To member
Inverse Property	**subTypes**

symbols—Symbols

Lists string symbol definitions. Symbols correspond to OpenVMS logical names or UNIX environment variables. Each list entry is a string of the form *name=value.*

Summary

Symbolic Name	
C binding	MCS_prop_symbols
OpenVMS binding	MCS$r_prop_symbols
Tag	NAD$K_ATT_SYMBOLS
Defined By	PERSISTENT_PROCESS
Required With **new**	No
Type	Normal
Data Type	MCS_LIST
Access	Read/write

tag—Type Tag

Contains a number that uniquely identifies a type. This value is normally generated by Oracle CDD/Repository, but an application can place its facility code in the first word of the value to distinguish types it creates from all other types.

Summary

Symbolic Name	
C binding	MCS_prop_tag
OpenVMS binding	MCS$r_prop_tag
Tag	NAD$K_ATT_TAG
Defined By	TYPE
Required With **new**	No
Type	Normal
Data Type	MCS_LONGINT
Access	Write-once-at-creation

toolName—Tool Name

Names the tool that was running when an EVENT record was created.

Summary

Symbolic Name	
C binding	MCS_prop_toolName
OpenVMS binding	MCS$r_prop_toolName
Tag	NAD$K_ATT_HISTORY_PRODUCT
Defined By	EVENT
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Read/write

toolVersion—Tool Version

Indicates the version number of the processor that executed as a result of this method invocation.

Summary

Symbolic Name	
C binding	MCS_prop_toolVersion
OpenVMS binding	MCS$r_prop_toolVersion
Tag	NAD$K_ATT_TOOLVERSION
Defined By	METHOD_INVOCATION
Required With **new**	No
Type	Normal
Data Type	MCS_STRING
Access	Write-once-at-creation

top—Top of Configuration Hierarchy

Identifies the VERSION (usually a COLLECTION) element that is the root of a context's configuration hierarchy.

Changing the value of this property with **setProp** results in the deletion of subdirectories under the context directory, and the creation of a new subdirectory structure to correspond to the new configuration hierarchy. The value of **top** cannot be changed if the context has open or reserved files.

top

Summary

Symbolic Name	
C binding	MCS_prop_top
OpenVMS binding	MCS$r_prop_top
Defined By	CONTEXT
Required With **new**	No
Type	Relation
Data Type	MCS_ELEMENTID
Access	Read/write
Relation Traversed	HAS_TOP_COLLECTION
Traversal Direction	To member
Inverse Property	**contextHavingAsTop**

---

userName—User Name

Names the user who sent a message to an element, resulting in the EVENT element possessing the property.

Summary

Symbolic Name	
C binding	MCS_prop_userName
OpenVMS binding	MCS$r_prop_userName
Tag	NAD$K_ATT_USERNAME
Defined By	EVENT
Required With **new**	Yes
Type	Normal
Data Type	MCS_STRING
Access	Write-once-at-creation

validationAction—Validation Action

Indicates the action to take if a validation fails.

Summary

Symbolic Name	
C binding	MCS_prop_validationAction
OpenVMS binding	MCS$r_prop_validationAction
Tag	NAD$K_ATT_VAL_ACTION
Defined By	VALIDATION
Required With **new**	No
Type	Normal
Data Type	MCS_SMALLINT
Access	Read/write

Constant Values

Value	Meaning
nad$k_warn	If validation fails, place a message on the error stack but continue the operation.
nad$k_fail	If validation fails, the operation fails.

The values of the **attachmentInContext** property are mutually exclusive.

validationApply—Operations to Validate

Indicates the operations to which the validation applies. The values can be combined to specify that the validation applies in more than one situation.

validationApply

Summary

Symbolic Name	
C binding	MCS_prop_validationApply
OpenVMS binding	MCS$r_prop_validationApply
Tag	NAD$K_ATT_VAL_APPLY
Defined By	VALIDATION
Required With **new**	No
Type	Normal
Data type	MCS_SMALLINT
Access	Read/write

Constant Values

Value	Meaning
nad$k_val_new	Apply when creating an object.
nad$k_val_setprop	Apply when modifying an object.
nad$k_val_free	Apply when deleting an object.
nad$k_val_reserve	Apply when reserving an object.
nad$k_val_replace	Apply when replacing an object.
nad$k_val_new_prot	Apply when creating a new type.
nad$k_val_setprop_prop	Apply when modifying a type.

validationQuery—Query Buffer for Validation

Allows a validation to be defined through a query buffer. (Supplied for compatibility with Oracle CDD/Repository V4.)

Summary

Symbolic Name	
C binding	MCS_prop_validationQuery
OpenVMS binding	MCS$r_prop_validationQuery
Tag	NAD$K_ATT_VAL_QUERY
Defined By	VALIDATION
Required With **new**	No
Type	Normal
Data type	MCS_MEMBLOCK
Access	Read/write

validationWhen—When to Validate

Indicates when the associated validation executes. The values can be combined to specify that the validation should run in more than one situation.

Summary

Symbolic Name	
C binding	MCS_prop_validationWhen
OpenVMS binding	MCS$r_prop_validationWhen
Tag	NAD$K_ATT_VAL_WHEN
Defined By	VALIDATION
Required With **new**	No
Type	Normal
Data type	MCS_SMALLINT
Access	Read/write

validationWhen

Constant Values

Value	Meaning
nad$k_val_start	Run validation before operation.
nad$k_val_end	Run validation after operation.
nad$k_val_ci	Run validation only if call is coming from the CI.
nad$k_val_mcs	Run validation only if call is coming from the MCS interface.
nad$k_val_ci_mcs	Run validation if call is either from the CI or the MCS interface.

versionable—Versionable Type

Indicates whether an element type is a subtype of VERSION.

Summary

Symbolic Name	
C binding	MCS_prop_versionable
OpenVMS binding	MCS$r_prop_versionable
Defined By	ELEMENT_TYPE
Required With **new**	No
Type	Computed
Data Type	MCS_BOOLEAN
Access	Read-only

Constant Values

Value	Meaning
TRUE	The element type is a subtype of VERSION.
FALSE	The element type is not a subtype of VERSION.

versionNum—Version Number

Contains the version number of a VERSION element.

Summary

Symbolic Name	
C binding	MCS_prop_versionNum
OpenVMS binding	MCS$r_prop_versionNum
Defined By	VERSION
Required With **new**	No
Type	Computed
Data Type	MCS_LONGINT
Access	Read-only

A
Relation Properties

This appendix contains summary information about relation properties. Many, but not all, of the properties listed here are described in more detail in Chapter 3; these properties are identified in the tables. Those properties listed here that are not described in Chapter 3 are defined by Oracle CDD/Repository for purposes of completeness.

A.1 Relation Properties by Relation Type

Table A–1 consists of an alphabetical listing of all relation types, accompanied by the names of the relation properties that traverse each type. Relation properties may be defined for each of the four directions:

To Owner
To Member
To All Owners
To All Members

Table A–1 Relation Properties by Relation Type

Relation	Direction			
Type	**To Owner**	**To Member**	**To All Owners**	**To All Members**
COMPOSITE_ PART	**hasParents**[1]	**hasChildren**[1]	-none-	**allChildren**[1]
DEPENDS_ON	**dependencies**[1]	**dependents**[1]	**allDependencies**[1]	**allDependents**[1]
HAS_COMPUTED_ PROPERTY	**typesHaving-CompProp**	**compPropDef**[1]	-none-	-none-
HAS_CONTEXT	**ppForContext**	**currContext**[1]	-none-	-none-

[1]A description of this property appears in Chapter 3.

(continued on next page)

Table A–1 (Cont.) Relation Properties by Relation Type

Relation Type	Direction			
	To Owner	To Member	To All Owners	To All Members
HAS_CURR_COLLECTION	**ppForCollection**	**currCollection**[1]	-none-	-none-
HAS_DATATYPE	**dataTypeUsers**	**dataType**[1]	-none-	-none-
HAS_DEFAULT_METHOD	**typesUsingMethod**	**methods**[1]	-none-	-none-
HAS_MESSAGE	**implementingMethods**	**implementsMessage**[1]	-none-	-none-
HAS_METHOD_USED[2]	**miUsingMethod**	**methodUsed**[1]	-none-	-none-
HAS_MSGARG	**messagesHaving-Msgarg**	**argSpec**[1]	-none-	-none-
HAS_MSG_SENT[2]	**miForMessage**	**msgSent**[1]	-none-	-none-
HAS_MSG_TARGET[2]	**miForTarget**	**msgTarget**[1]	-none-	-none-
HAS_PARENT	**childPartitions**[1]	**basePartition**[1] **parentPartition**[1]	**allChildPartitions**[1]	**allParentPartitions**[1]
HAS_POSTAMBLE	**methodUsing-Postamble**	**postamble**[1]	-none-	-none-
HAS_PREAMBLE	**methodUsing-Preamble**	**preamble**[1]	-none-	-none-
HAS_PROPERTY	**typesHavingProp**	**propDef**[1]	-none-	-none-
HAS_RELATED_PARTITION	**partitionsRelatedTo**	**related**[1]	-none-	-none-
HAS_RELATION	**legalOwners**[1]	**ownsRelation**[1]	-none-	-none-
HAS_RELATION_PROPERTY	**typesHavingRelProp**	**relPropDef**[1]	-none-	-none-
HAS_SUPERTYPE	**subTypes**[1]	**superTypes**[1]	**allSubTypes**[1]	**allSuperTypes**[1]
HAS_TOP_COLLECTION	**contextHavingAsTop**	**top**[1]	-none-	-none-
HISTORY_LIST[2]	**historyOwner**	**history**[1]	-none-	-none-
IMPLEMENTS_METHOD	**compPropUsing**	**implementsMethod**[1]	-none-	-none-

[1]A description of this property appears in Chapter 3.

[2]This relation type is not documented in Chapter 1.

(continued on next page)

Table A–1 (Cont.) Relation Properties by Relation Type

Relation	Direction			
Type	**To Owner**	**To Member**	**To All Owners**	**To All Members**
IMPLEMENTS_ RELATION	**relPropUsing**	**implementsRelation**[1]	-none-	-none-
INVOKES_TOOL	**invokedBy**	**invokes**[1]	-none-	-none-
METHOD_INPUT[2]	**derives**[1]	**derivedFrom**[1]	-none-	-none-
METHOD_ OUTPUT[2]	**derives**[1]	**derivedFrom**[1]	-none-	-none-
METHOD_ PARAMETER[2]	-none-	-none-	**allDerives**[1]	**allDerivedFrom**[1]
OBJECT_ VALIDATION[2]	**validationForType**	**associatedValidations**[1]	-none-	-none-
OPENED_BY	**openedBy**[1]	**openedFiles**[1]	-none-	-none-
RELATION[3]	-none-	-none-	-none-	-none-
RELATION_ MEMBER	**relationMember**[1]	**legalMembers**[1]	-none-	-none-
RESERVED_BY	**checkout**[1]	**reservedBy**	-none-	-none-

[1]A description of this property appears in Chapter 3.

[2]This relation type is not documented in Chapter 1.

[3]The **relMember** and **relOwner** properties use the RELATION relation type but are normal properties, not relation properties.

A.2 Relation Properties by Property Name

Table A–2 consists of an alphabetical listing of all relation and closure properties, accompanied by the following information:

- the element type that defines the property
- the relation type traversed by the property
- the traversal direction
- the data type
- the access type

- the inverse property

Table A–2 Relation Properties by Property Name

Property	Defined By	Traverses	To	Data Type/ Access Type	Inverse
allChildPartitions[1]	PARTITION	HAS_PARENT	AO	S/RO	**allParentPartitions**[1]
allChildren[1]	COMPOSITE	COMPOSITE_ PART	AM	S/RO	-none-
allDependencies[1]	VERSION	DEPENDS_ON	AO	S/RO	**allDependents**[1]
allDependents[1]	VERSION	DEPENDS_ON	AM	S/RO	**allDependencies**[1]
allDerivedFrom[1]	VERSION	METHOD_ PARAMETER	AM	S/RO	**allDerives**[1]
allDerives[1]	VERSION	METHOD_ PARAMETER	AO	S/RO	**allDerivedFrom**[1]
allParentPartitions[1]	PARTITION	HAS_PARENT	AM	S/RO	**allChildPartitions**[1]
allSubTypes[1]	TYPE	HAS_SUPERTYPE	AO	S/RO	**allSuperTypes**[1]
allSuperTypes[1]	TYPE	HAS_SUPERTYPE	AM	S/RO	**allSubTypes**[1]
argSpec[1]	MESSAGE	HAS_MSGARG	M	S/RW	**messagesHavingMsgarg**
associatedValidations[1]	ELEMENT_TYPE	OBJECT_ VALIDATION[2]	M	S/RW	**validationForType**
basePartition[1]	CONTEXT	HAS_PARENT	M	E/WOC	-none-
checkout[1]	CONTEXT	RESERVED_BY	O	S/RO	**reservedBy**
childPartitions[1]	PARTITION	HAS_PARENT	O	S/RO	**parentPartitions**[1]
compPropDef[1]	ELEMENT_TYPE	HAS_COMPUTED_ PROPERTY	M	S/RW	**typesHavingCompProp**
compPropUsing	METHOD	IMPLEMENTS_ METHOD	O	E/RW	**implementsMethod**[1]
contextHavingAsTop	VERSION	HAS_TOP_ COLLECTION	O	E/RW	**top**[1]
currCollection[1]	PERSISTENT_ PROCESS	HAS_CURR_ COLLECTION	M	E/RO	**ppForCollection**

[1]A description of this property appears in Chapter 3.

[2]This relation type is not documented in Chapter 1.

Traversal Directions: O—To owner; M—To member; AO—To all owners; AM—To all members.
Data Types: S—MCS_SCAN; E—MCS_ELEMENTID.
Access Types: RO—Read-only; RW—Read/Write; WO—Write once; WOC—Write once at creation.

(continued on next page)

Table A–2 (Cont.) Relation Properties by Property Name

Property	Defined By	Traverses	To	Data Type/ Access Type	Inverse
currContext[1]	PERSISTENT_ PROCESS	HAS_CONTEXT	M	E/RW	**ppForContext**
dataType[1]	PROPERTY_TYPE MSGARG	HAS_DATATYPE	M	E/WOC	**dataTypeUsers**
dataTypeUsers	DATA_TYPE	HAS_DATATYPE	O	S/RO	**dataType**[1]
dependencies[1]	VERSION	DEPENDS_ON	O	S/RW	**dependents**[1]
dependents[1]	VERSION	DEPENDS_ON	M	S/RW	**dependencies**[1]
derivedFrom[1]	VERSION METHOD_ INVOCATION	METHOD_ OUTPUT[2] METHOD_INPUT[2]	M	S/WOC	**derives**[1]
derives[1]	VERSION METHOD_ INVOCATION	METHOD_INPUT[2] METHOD_ OUTPUT[2]	O	S/WOC	**derivedFrom**[1]
hasChildren[1]	COMPOSITE	COMPOSITE_ PART	M	S/RO	**hasParents**[1]
hasParents[1]	VERSION	COMPOSITE_ PART	O	S/RO	**hasChildren**[1]
history[1]	NAMED_ ELEMENT	HISTORY_LIST[2]	M	S/RW	**historyOwner**
historyOwner	EVENT	HISTORY_LIST[2]	O	E/RW	**history**[1]
implementingMethods	MESSAGE	HAS_MESSAGE	O	S/RW	**implementsMessage**[1]
implementsMessage[1]	METHOD	HAS_MESSAGE	M	E/RW	**implementingMethods**
implementsMethod[1]	HAS_COMPUTED_ PROPERTY	IMPLEMENTS_ METHOD	M	E/RW	**compPropUsing**
implementsRelation[1]	HAS_RELATION_ PROPERTY	IMPLEMENTS_ RELATION	M	E/RW	**relPropUsing**
invokedBy	BINARY_TOOL TEXT_TOOL	INVOKES_TOOL	O	S/RW	**invokes**[1]
invokes[1]	TOOL	INVOKES_TOOL	M	E/RW	**invokedBy**

[1]A description of this property appears in Chapter 3.

[2]This relation type is not documented in Chapter 1.

Traversal Directions: O—To owner; M—To member; AO—To all owners; AM—To all members.
Data Types: S—MCS_SCAN; E—MCS_ELEMENTID.
Access Types: RO—Read-only; RW—Read/Write; WO—Write once; WOC—Write once at creation.

(continued on next page)

Table A–2 (Cont.) Relation Properties by Property Name

Property	Defined By	Traverses	To	Data Type/ Access Type	Inverse
legalMembers[1]	RELATION_TYPE	RELATION_ MEMBER	M	S/RW	**relationMember**[1]
legalOwners[1]	RELATION_TYPE	HAS_RELATION	O	S/RW	**ownsRelation**[1]
messagesHavingMsgarg	MSGARG	HAS_MSGARG	O	S/RW	**argSpec**[1]
methods[1]	ELEMENT_TYPE	HAS_DEFAULT_ METHOD	M	S/RW	**typesUsingMethod**
methodUsed[1]	ATIS_METHOD_ INVOC	HAS_METHOD_ USED[2]	M	E/WOC	**miUsingMethod**
methodUsingPostamble	METHOD	HAS_POSTAMBLE	O	S/RW	**postamble**[1]
methodUsingPreamble	METHOD	HAS_PREAMBLE	O	S/RW	**preamble**[1]
miForMessage	MESSAGE	HAS_MSG_SENT[2]	O	S/RW	**msgSent**[1]
miForTarget	VERSION	HAS_MSG_ TARGET[2]	O	S/RO	**msgTarget**[1]
miUsingMethod	METHOD	HAS_METHOD_ USED[2]	O	S/RW	**methodUsed**[1]
msgSent[1]	ATIS_METHOD_ INVOC	HAS_MSG_SENT[2]	M	E/WOC	**miForMessage**
msgTarget[1]	ATIS_METHOD_ INVOC	HAS_MSG_ TARGET[2]	M	E/WOC	**miForTarget**
openedBy[1]	CONTEXT	OPENED_BY	O	S/RO	**openedFiles**[1]
openedFiles[1]	CONTEXT	OPENED_BY	M	S/RO	**openedBy**[1]
ownsRelation[1]	ELEMENT_TYPE	HAS_RELATION	M	S/RW	**legalOwners**[1]
parentPartition[1]	PARTITION	HAS_PARENT	M	E/WOC	**childPartitions**[1]
partitionsRelatedTo	PARTITION	HAS_RELATED_ PARTITION	O	S/RO	**related**[1]
postamble[1]	METHOD	HAS_POSTAMBLE	M	S/RW	**methodUsingPostamble**
preamble[1]	METHOD	HAS_PREAMBLE	M	S/RW	**methodUsingPreamble**

[1]A description of this property appears in Chapter 3.

[2]This relation type is not documented in Chapter 1.

Traversal Directions: O—To owner; M—To member; AO—To all owners; AM—To all members.
Data Types: S—MCS_SCAN; E—MCS_ELEMENTID.
Access Types: RO—Read-only; RW—Read/Write; WO—Write once; WOC—Write once at creation.

(continued on next page)

Table A–2 (Cont.) Relation Properties by Property Name

Property	Defined By	Traverses	To	Data Type/ Access Type	Inverse
ppForCollection	COLLECTION	HAS_CURR_ COLLECTION	O	E/RO	**currCollection**[1]
ppForContext	CONTEXT	HAS_CONTEXT	O	E/RW	**currContext**[1]
propDef[1]	ELEMENT_TYPE	HAS_PROPERTY	M	S/RW	**typesHavingProp**
related[1]	PARTITION	HAS_RELATED_ PARTITION	M	S/RW	**partitionsRelatedTo**
relationMember[1]	ELEMENT_TYPE	RELATION_ MEMBER	O	S/RW	**legalMembers**[1]
relPropDef[1]	ELEMENT_TYPE	HAS_RELATION_ PROPERTY	M	S/RW	**typesHavingRelProp**
relPropUsing	RELATION_TYPE	IMPLEMENTS_ RELATION	O	E/RW	**implementsRelation**[1]
reservedBy	VERSION	RESERVED_BY	M	E/RO	**checkout**[1]
subTypes[1]	TYPE	HAS_SUPER_ TYPE	O	S/RO	**superTypes**[1]
superTypes[1]	ELEMENT_TYPE	HAS_SUPER_ TYPE	M	S/WOC	**subTypes**[1]
top[1]	CONTEXT	HAS_TOP_ COLLECTION	M	E/RW	**contextHavingAsTop**
typesHavingCompProp	PROPERTY_TYPE	HAS_COMPUTED_ PROPERTY	O	S/RW	**compPropDef**[1]
typesHavingProp	PROPERTY_TYPE	HAS_PROPERTY	O	S/RW	**propDef**[1]
typesHavingRelProp	PROPERTY_TYPE	HAS_RELATION_ PROPERTY	O	S/RW	**relPropDef**[1]
typesUsingMethod	METHOD	HAS_DEFAULT_ METHOD	O	S/RW	**methods**[1]
validationForType	VALIDATION	OBJECT_ VALIDATION	O	E/RW	**associatedValidations**[1]

[1]A description of this property appears in Chapter 3.

Traversal Directions: O—To owner; M—To member; AO—To all owners; AM—To all members.
Data Types: S—MCS_SCAN; E—MCS_ELEMENTID.
Access Types: RO—Read-only; RW—Read/Write; WO—Write once; WOC—Write once at creation.

B

Implemented Information Model Diagrams

This appendix contains implemented information model (IIM) diagrams for many of the element types and relation types described in this manual. These diagrams show how instances of element types can associate with instances of other (or the same) types. For example, Figure B–2 shows that PERSISTENT_PROCESS elements associate with CONTEXT elements using HAS_CONTEXT relationships. In the object-oriented view of the Oracle CDD/Repository schema, these associations appear as relation properties, such as **currContext** on PERSISTENT_PROCESS elements.

Figure B–1 lists the IIM diagram conventions.

Figure B–1 IIM Diagram Conventions

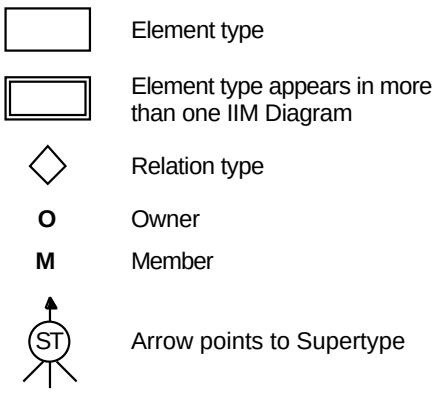


Figure B–2 illustrates the top of the element type hierarchy.

Figure B–2 Implemented Information Model: ELEMENT

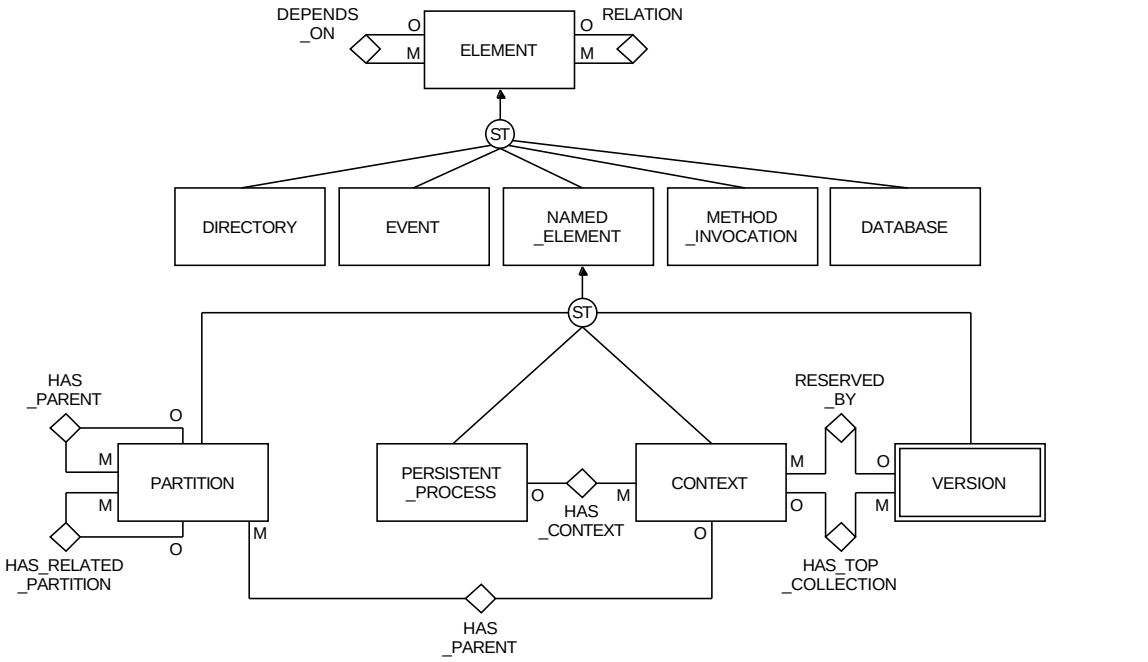


Figure B–3 illustrates the versionable element types (other than subtypes of AGGREGATE) including the metadata types.

Figure B–3 Implemented Information Model: VERSION

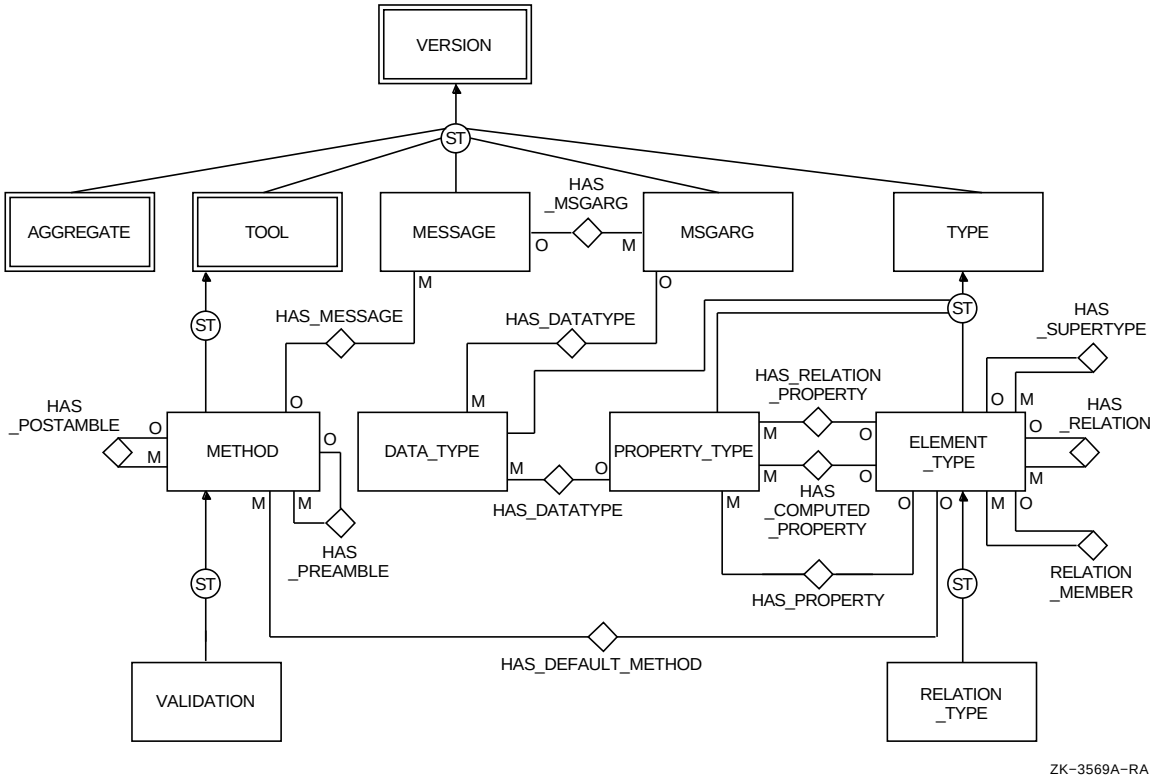
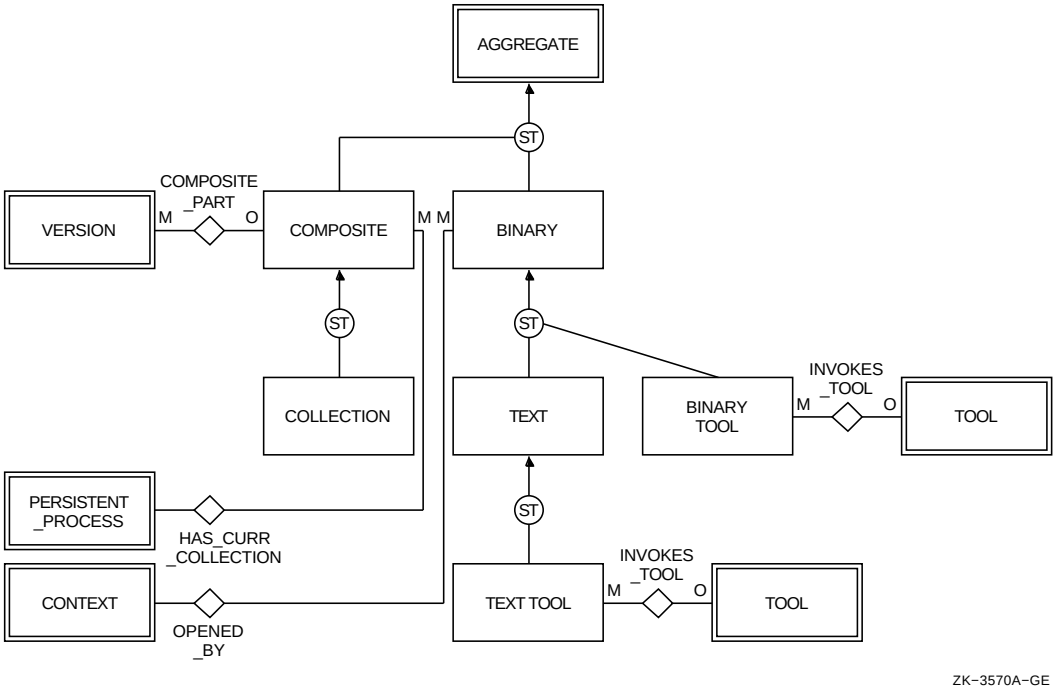


Figure B–4 illustrates AGGREGATE and subtypes.

Figure B–4 Implemented Information Model: AGGREGATE



Index

A

E

F

G

P

R

S

T

U

V