Contents

Send Us Your Comments .................................................. xi

Preface ................................................................. xiii

1 Element Type Descriptions

ACAS_METHOD_INVOC .................................................. 1–5
AGGREGATE ............................................................... 1–7
ATIS_METHOD_INVOC .................................................. 1–10
BINARY ................................................................. 1–12
BINARY_TOOL ........................................................... 1–15
COLLECTION ............................................................ 1–18
COMPOSITE ............................................................ 1–21
COMPOSITE_PART ..................................................... 1–24
CONTEXT ............................................................... 1–26
DATABASE .............................................................. 1–28
DATA_TYPE .............................................................. 1–30
DEPENDS_ON ........................................................... 1–33
DIRECTORY .............................................................. 1–35
ELEMENT ............................................................... 1–37
ELEMENT_TYPE ......................................................... 1–38
EVENT ................................................................. 1–41
HAS_COMPUTED_PROPERTY .......................................... 1–43
HAS_CONTEXT ........................................................... 1–45
HAS_CURR_COLLECTION ............................................ 1–47
HAS_DATATYPE .......................................................... 1–49
HAS_DEFAULT_METHOD .............................................. 1–51
HAS_MESSAGE .......................................................... 1–53
HAS_MSGARG ............................................................ 1–55
<table>
<thead>
<tr>
<th>Term</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAS_PARENT</td>
<td>1–57</td>
</tr>
<tr>
<td>HAS_POSTamble</td>
<td>1–59</td>
</tr>
<tr>
<td>HAS_PREAMBLE</td>
<td>1–61</td>
</tr>
<tr>
<td>HAS_PROPERTY</td>
<td>1–63</td>
</tr>
<tr>
<td>HASRELATED_PARTITION</td>
<td>1–65</td>
</tr>
<tr>
<td>HAS_RELATION</td>
<td>1–67</td>
</tr>
<tr>
<td>HAS_RELATION_PROPERTY</td>
<td>1–69</td>
</tr>
<tr>
<td>HAS_SUPERTYPE</td>
<td>1–71</td>
</tr>
<tr>
<td>HAS_TOP_COLLECTION</td>
<td>1–73</td>
</tr>
<tr>
<td>IMPLEMENTS_METHOD</td>
<td>1–75</td>
</tr>
<tr>
<td>IMPLEMENTS_RELATION</td>
<td>1–77</td>
</tr>
<tr>
<td>INVOKES_TOOL</td>
<td>1–79</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>1–81</td>
</tr>
<tr>
<td>METHOD</td>
<td>1–84</td>
</tr>
<tr>
<td>METHOD_INVOCATION</td>
<td>1–87</td>
</tr>
<tr>
<td>MSGARG</td>
<td>1–89</td>
</tr>
<tr>
<td>NAMED_ELEMENT</td>
<td>1–92</td>
</tr>
<tr>
<td>OPENED_BY</td>
<td>1–94</td>
</tr>
<tr>
<td>PARTITION</td>
<td>1–96</td>
</tr>
<tr>
<td>PERSISTENT_PROCESS</td>
<td>1–98</td>
</tr>
<tr>
<td>PROPERTY_TYPE</td>
<td>1–100</td>
</tr>
<tr>
<td>RELATION</td>
<td>1–103</td>
</tr>
<tr>
<td>RELATION_MEMBER</td>
<td>1–105</td>
</tr>
<tr>
<td>RELATION_TYPE</td>
<td>1–107</td>
</tr>
<tr>
<td>RESERVED_BY</td>
<td>1–110</td>
</tr>
<tr>
<td>TEXT</td>
<td>1–112</td>
</tr>
<tr>
<td>TEXT_TOOL</td>
<td>1–115</td>
</tr>
<tr>
<td>TOOL</td>
<td>1–118</td>
</tr>
<tr>
<td>TYPE</td>
<td>1–121</td>
</tr>
<tr>
<td>VALIDATION</td>
<td>1–124</td>
</tr>
<tr>
<td>VERSION</td>
<td>1–127</td>
</tr>
</tbody>
</table>
2 Message Descriptions

attach .................................................. 2–5
build .................................................... 2–7
close ................................................... 2–9
control ................................................ 2–12
detach .................................................. 2–15
differences ........................................... 2–18
edit ...................................................... 2–20
export .................................................. 2–22
free ..................................................... 2–24
freeze .................................................. 2–29
getProp ................................................ 2–31
import .................................................. 2–34
merge ................................................... 2–36
new ..................................................... 2–42
open .................................................... 2–52
promote ................................................ 2–55
purge ................................................... 2–59
rename .................................................. 2–61
replace ............................................... 2–63
reserve ............................................... 2–68
setProp ............................................... 2–74
translate ............................................ 2–79
unfreeze ............................................. 2–81
unreserve ........................................... 2–83
update ............................................... 2–87
verify .................................................. 2–89

3 Property Descriptions

access .................................................. 3–3
accessType .......................................... 3–5
aliases ............................................... 3–5
allCheckouts ...................................... 3–6
allChildPartitions ................................. 3–7
allChildren ........................................ 3–7
allDependencies ................................... 3–8
allDependents ............................................. 3–9
allDerivedFrom ........................................... 3–9
allDerives .............................................. 3–10
allElementTypes ......................................... 3–11
allInstances ............................................ 3–11
allParentPartitions ..................................... 3–12
allSubTypes ............................................ 3–12
allSuperTypes .......................................... 3–13
alternateNames ......................................... 3–14
application ............................................. 3–14
argList ................................................ 3–15
argSpec ................................................. 3–15
argsSent ................................................. 3–15
associatedValidations .................................. 3–16
attachment ............................................. 3–17
attachmentInContext .................................. 3–17
autopurge .............................................. 3–18
availVersion ......................................... 3–19
basePartition ......................................... 3–20
baseType ............................................. 3–21
baseTypeSize ......................................... 3–22
branchName ........................................ 3–22
checkout .............................................. 3–23
childPartitions ...................................... 3–23
compPropDef ......................................... 3–24
contextDir .......................................... 3–25
contextName ......................................... 3–25
controlled .......................................... 3–26
CPUTime ............................................... 3–26
createdDate ......................................... 3–27
currCollection ...................................... 3–28
currContext ........................................ 3–28
dataType ............................................. 3–29
defaultAccess ..................................... 3–30
defaultAttachment .................................. 3–31
definedLegalMembers ................................. 3–31
definedLegalOwners ................................. 3–32
msgSent .................................................. 3–57
msgTarget .................................................. 3–57
mutable .................................................. 3–58
name .................................................... 3–59
nextVersions .............................................. 3–60
node ..................................................... 3–60
numChildren .............................................. 3–61
openedBy .............................................. 3–62
openedFiles .............................................. 3–62
optionsString .............................................. 3–63
OSVersion .............................................. 3–64
owner .................................................... 3–64
ownsRelation .............................................. 3–65
parentInContext ........................................ 3–65
parentPartition ........................................ 3–66
partitionDir ............................................. 3–66
passingMechanism ........................................ 3–67
path ..................................................... 3–68
pattern ............................................... 3–68
postamble .............................................. 3–69
preamble ............................................... 3–69
prevVersions ........................................... 3–70
processingName ........................................ 3–71
propDef ................................................ 3–71
referenceCount ......................................... 3–72
related ................................................ 3–73
relationMember ........................................ 3–73
relMember ............................................ 3–74
relOwner ............................................. 3–75
relPropDef ............................................ 3–75
required ............................................... 3–76
rootBranchInstances ................................... 3–77
rootBranchName ........................................ 3–77
rootPath .............................................. 3–78
rootVersion ........................................... 3–79
scale .................................................. 3–79
scalingFactor ........................................ 3–80
<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Documentation Conventions</td>
<td>xv</td>
</tr>
<tr>
<td>3–1</td>
<td>Privilege Bits for Access Control Lists</td>
<td>3–4</td>
</tr>
<tr>
<td>A–1</td>
<td>Relation Properties by Relation Type</td>
<td>A–1</td>
</tr>
<tr>
<td>A–2</td>
<td>Relation Properties by Property Name</td>
<td>A–4</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>italic type</td>
<td>Italics emphasize important information, indicate variables, and indicate complete titles of manuals.</td>
</tr>
<tr>
<td>UPPERCASE</td>
<td>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.</td>
</tr>
<tr>
<td>NAMED_ELEMENT</td>
<td>The names of element types are set in small capitals.</td>
</tr>
<tr>
<td>merge</td>
<td>The names of elements, messages, and properties are set in bold type.</td>
</tr>
<tr>
<td>hasChildren</td>
<td></td>
</tr>
<tr>
<td>list_value</td>
<td>References to values you supply, such as arguments, are set in italics.</td>
</tr>
<tr>
<td>0 ELEMENT</td>
<td></td>
</tr>
<tr>
<td>1 NAMED_ELEMENT</td>
<td>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.</td>
</tr>
<tr>
<td>2 VERSION</td>
<td></td>
</tr>
</tbody>
</table>
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

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

(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 HASRELATED_PARTITION
2 HAS_TOP_COLLECTION
2 OPENED_BY
2 RESERVED_BY
ACAS_METHOD_INVOC

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$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 (ELEMENT)
- 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)
AGGREGATE

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

Methods Not Allowed

-None-
ATIS_METHOD_INVOC

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)
  CPUPTime (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

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)
one (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)
- allDerived (VERSION)
- allElementTypeDeps (ELEMENT)
- alternateNames (NAMED_ELEMENT)
- attachmentInContext (VERSION)
- availVersion (VERSION)
- branchName (VERSION)
- controlled (VERSION)
- createDate (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)
extport (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—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)
createdDate (NAMED_ELEMENT)
dependencies (VERSION)
COLLECTION

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

color (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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_elm_composite</td>
<td>MCS$r_elm_composite</td>
</tr>
</tbody>
</table>

Hierarchy Level

<table>
<thead>
<tr>
<th>0 ELEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 NAMED_ELEMENT</td>
</tr>
<tr>
<td>2 VERSION</td>
</tr>
<tr>
<td>3 AGGREGATE</td>
</tr>
<tr>
<td>4 COMPOSITE</td>
</tr>
</tbody>
</table>

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—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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>MCS_elm_composite_part</th>
<th>MCS$r_elm_composite_part</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy Level</td>
<td>0 ELEMENT</td>
<td></td>
<td>1 RELATION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 DEPENDS_ON</td>
<td></td>
<td>3 COMPOSITE_PART</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>NAD$K_REL_COMPOSITE_PART</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Defined Properties

attachment

Inherited Properties

allElementType (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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>COMPOSITE</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>VERSION</td>
<td>true</td>
</tr>
</tbody>
</table>
CONTEXT

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

<table>
<thead>
<tr>
<th>C binding</th>
<th>MCS_elm_context</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenVMS binding</td>
<td>MCS$r_elm_context</td>
</tr>
</tbody>
</table>

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—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-
DATABASE

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)
DATA_TYPE

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>ELEMENT</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>ELEMENT</td>
<td>true</td>
</tr>
</tbody>
</table>
DIRECTORY

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_elm_element</td>
<td>MCS$r_elm_element</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hierarchy Level</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ELEMENT</td>
<td>NAD$K_ENT_ELEMENT</td>
</tr>
</tbody>
</table>

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—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

OpenVMS binding

MCS_elm_element_type

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)
ELEMENT_TYPE

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

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
OpenVMS binding

MCS_elm_event
MCS$r_elm_event

Hierarchy Level
0 ELEMENT
1 EVENT

Tag
NAD$K_ENT_HISTORY

Defined Properties
argList
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—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

<table>
<thead>
<tr>
<th>C binding</th>
<th>MCS_elm_has_computed_property</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenVMS binding</td>
<td>MCS$r_elm_has_computed_property</td>
</tr>
</tbody>
</table>

Hierarchy Level

<table>
<thead>
<tr>
<th>0 ELEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 RELATION</td>
</tr>
<tr>
<td>2 DEPENDS_ON</td>
</tr>
<tr>
<td>3 HASPROPERTY</td>
</tr>
<tr>
<td>4 HAS_COMPUTED_PROPERTY</td>
</tr>
</tbody>
</table>

Tag

NAD$K_REL_HAS_COMP_PROP

Defined Properties

- implementsMethod—Required

Inherited Properties

- allElementTypes (ELEMENT)
- elementType (ELEMENT)
- mutable (HASPROPERTY)
- relMember—Required (RELATION)
- relOwner—Required (RELATION)
- required—Required (HASPROPERTY)

Defined Methods

- none-

Refined Methods

- none-

Inherited Methods

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

Methods Not Allowed

-none-

Participants

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>ELEMENT_TYPE</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>PROPERTY_TYPE</td>
<td>true</td>
</tr>
</tbody>
</table>
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-
### Participants

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>PERSISTENT_PROCESS</td>
<td>false</td>
</tr>
<tr>
<td>member</td>
<td>CONTEXT</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_CURR_COLLECTION

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**

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>PERSISTENT_PROCESS</td>
<td>false</td>
</tr>
<tr>
<td>member</td>
<td>COMPOSITE</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_DATATYPE

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
OpenVMS binding

MCS_elm_has_datatype
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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>PROPERTY_TYPE, MSGARG</td>
<td>false</td>
</tr>
<tr>
<td>member</td>
<td>DATA_TYPE</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_DEFAULT_METHOD

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
- OpenVMS binding

MCS_elm_has_default_method
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**

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>ELEMENT_TYPE</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>METHOD</td>
<td>false</td>
</tr>
</tbody>
</table>
HAS_MESSAGE

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>METHOD</td>
<td>false</td>
</tr>
<tr>
<td>member</td>
<td>MESSAGE</td>
<td>true</td>
</tr>
</tbody>
</table>
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
OpenVMS binding

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>MESSAGE</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>MSGARG</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_PARENT

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

<table>
<thead>
<tr>
<th>C binding</th>
<th>C binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_elm_has_parent</td>
<td>MCS$r_elm_has_parent</td>
</tr>
</tbody>
</table>

OpenVMS binding

<table>
<thead>
<tr>
<th>Hierarchy Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ELEMENT</td>
</tr>
<tr>
<td>1 RELATION</td>
</tr>
<tr>
<td>2 HAS_PARENT</td>
</tr>
</tbody>
</table>

Tag

NAD$K_REL_PARENT_P

Defined Properties

-none-

Inherited Properties

-none-

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>PARTITION</td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>CONTEXT</td>
<td></td>
</tr>
<tr>
<td>member</td>
<td>PARTITION</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_POSTAMBLE

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-
### Participants

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>METHOD</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>METHOD</td>
<td>true</td>
</tr>
</tbody>
</table>
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-
### Participants

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>METHOD</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>METHOD</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_PROPERTY

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
OpenVMS binding

MCS_elm_has_property
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)
elemType (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)
HASPROPERTY

Methods Not Allowed

- none -

Participants

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>ELEMENT_TYPE</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>PROPERTY_TYPE</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_RELATED_PARTITION

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
OpenVMS binding

MCS_elm_has_related_partition
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**

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>PARTITION</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>PARTITION</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_RELATION

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>ELEMENT_TYPE</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>RELATION_TYPE</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_RELATION_PROPERTY

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_RELATIONPROPERTY

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>ELEMENT_TYPE</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>PROPERTY_TYPE</td>
<td>true</td>
</tr>
</tbody>
</table>
HAS_SUPERTYPE

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>TYPE</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>TYPE</td>
<td>true</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>CONTEXT</td>
<td>false</td>
</tr>
<tr>
<td>member</td>
<td>VERSION</td>
<td>true</td>
</tr>
</tbody>
</table>
IMPLEMENTS METHOD

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>HAS_COMPUTEDPROPERTY</td>
<td>false</td>
</tr>
<tr>
<td>member</td>
<td>METHOD</td>
<td>true</td>
</tr>
</tbody>
</table>
**IMPLEMENTS_RELATION**

**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_elm_implements_relation</td>
<td>MCS$r_elm_implements_relation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hierarchy Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ELEMENT</td>
</tr>
<tr>
<td>1 RELATION</td>
</tr>
<tr>
<td>2 DEPENDS_ON</td>
</tr>
<tr>
<td>3 IMPLEMENTS_RELATION</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAD$K_REL_IMP_RELATION</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>HAS_RELATION_</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>PROPERTY</td>
<td></td>
</tr>
<tr>
<td>member</td>
<td>RELATION_TYPE</td>
<td>true</td>
</tr>
</tbody>
</table>

1–78  Element Type Descriptions
INVOKES_TOOL

INVOKES_TOOL—Invokes Tool

Defines relationships that associate aTool 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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>TOOL</td>
<td>false</td>
</tr>
<tr>
<td>member</td>
<td>BINARY_TOOL</td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>TEXT_TOOL</td>
<td></td>
</tr>
</tbody>
</table>

1–80 Element Type Descriptions
MESSAGE

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
OpenVMS binding

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 (ELEMENT)
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—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
OpenVMS binding
MCS_elm_method
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—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

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)
built (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)
MSGARG

verify (ELEMENT)
Methods Not Allowed
-none-
NAMED_ELEMENT

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

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
- OpenVMS binding

MCS_elm_opened_by
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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>CONTEXT</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>BINARY</td>
<td>true</td>
</tr>
</tbody>
</table>
PARTITION

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

<table>
<thead>
<tr>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_elm_partition</td>
<td>MCS$r_elm_partition</td>
</tr>
</tbody>
</table>

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)
PARTITION

Defined Methods

-refine-

Refined Methods

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

Inherited Methods

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

Methods Not Allowed

-refine-
PERSISTENT_PROCESS

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
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_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)
PROPERTY_TYPE

description (NAMED_ELEMENT)
elemntType (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
- OpenVMS binding

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>ELEMENT</td>
<td>false</td>
</tr>
<tr>
<td>member</td>
<td>ELEMENT</td>
<td>false</td>
</tr>
</tbody>
</table>
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

-participants-

Participants

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>RELATION_TYPE</td>
<td>true</td>
</tr>
</tbody>
</table>
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)
built (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

---

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

<table>
<thead>
<tr>
<th>Participant</th>
<th>Legal Values</th>
<th>Many</th>
</tr>
</thead>
<tbody>
<tr>
<td>owner</td>
<td>VERSION</td>
<td>true</td>
</tr>
<tr>
<td>member</td>
<td>CONTEXT</td>
<td>false</td>
</tr>
</tbody>
</table>
TEXT

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          MCS(elm_text  MCS$r(elm_text
C binding               MCS_elm_text
OpenVMS binding         MCS$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)
freeendTime (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)
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
OpenVMS binding

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)
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)
- allElementType (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)
elemType (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—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)
VALIDATION

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
detach
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-
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
attach

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
6 TEXT_tool inherits
4 COMPOSITE inherits
5 COLLECTION inherits
3 MESSAGE inherits
3 MSGARG inherits
3 TOOL inherits
build

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
4 TEXT Tool 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 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—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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_TO_NONE</td>
<td>Control only this element. This is the default.</td>
</tr>
<tr>
<td>MCS_TO_BOTTOM</td>
<td>Control this element and all uncontrolled elements owned, either directly or indirectly, through dependency relationships. These include direct and indirect children of a composite.</td>
</tr>
</tbody>
</table>

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.
control

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
6 TEXT TOOL inherits
4 COMPOSITE inherits
5 COLLECTION refines
3 MESSAGE inherits
3 MSGARG inherits
3 TOOL inherits
4 METHOD 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.

**NOTATTACH**
- 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—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—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
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—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

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_TO_NONE</td>
<td>Free only this element. This is the default.</td>
</tr>
<tr>
<td>MCS_TO_BOTTOM</td>
<td>Free this element and all direct and indirect</td>
</tr>
<tr>
<td></td>
<td>children in the collection hierarchy.</td>
</tr>
</tbody>
</table>

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
 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

Message Descriptions 2–25
free

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

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ELEMENT</td>
<td>does not recognize</td>
</tr>
<tr>
<td>1 NAMED ELEMENT</td>
<td>does not recognize</td>
</tr>
<tr>
<td>2 VERSION</td>
<td><strong>defines</strong></td>
</tr>
<tr>
<td>3 AGGREGATE</td>
<td>inherits</td>
</tr>
<tr>
<td>4 BINARY</td>
<td>inherits</td>
</tr>
<tr>
<td>5 BINARY_TOOL</td>
<td>inherits</td>
</tr>
<tr>
<td>6 TEXT_TOOL</td>
<td>inherits</td>
</tr>
<tr>
<td>4 COMPOSITE</td>
<td>inherits</td>
</tr>
<tr>
<td>5 COLLECTION</td>
<td>inherits</td>
</tr>
<tr>
<td>3 MESSAGE</td>
<td>inherits</td>
</tr>
<tr>
<td>3 MSGARG</td>
<td>inherits</td>
</tr>
<tr>
<td>3 TOOL</td>
<td>inherits</td>
</tr>
<tr>
<td>4 METHOD</td>
<td>inherits</td>
</tr>
<tr>
<td>5 VALIDATION</td>
<td>inherits</td>
</tr>
<tr>
<td>3 TYPE</td>
<td>inherits</td>
</tr>
<tr>
<td>4 DATA_TYPE</td>
<td>inherits</td>
</tr>
<tr>
<td>4 ELEMENT_TYPE</td>
<td>inherits</td>
</tr>
</tbody>
</table>
freeze

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 DIRECTRY 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
getProp

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—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.
merge

**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)**
**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)**
**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)**
**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)**
**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.
merge

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_MERGE_CONFLICT</td>
<td>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.</td>
</tr>
<tr>
<td>MCS_MERGE_MERGE_CHANGED</td>
<td>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.</td>
</tr>
<tr>
<td>MCS_MERGE_SOURCE_CHANGED</td>
<td>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.</td>
</tr>
<tr>
<td>MCS_MERGE_UNCHANGED</td>
<td>The child is the same in the ancestor, source, and merge lists.</td>
</tr>
</tbody>
</table>

**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.
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

---

**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
new

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.


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 \texttt{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 \texttt{partitionDir} directory specifies an existing directory.

REQATTMIS No value was supplied for the \texttt{name} property.

\textbf{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 \texttt{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 \texttt{dataType} property and either the \texttt{CDD\$DATATYPE} or \texttt{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.

\textbf{RELATION}

1. Verify that each of the members specified for the \texttt{relMember} property is of a type allowed by the \texttt{legalMembers} property of the relation's type.

2. Verify that each of the owners specified for the \texttt{relOwner} property is of a type allowed by the \texttt{legalOwners} property of the relation's type.

new

**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.


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**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINARIALREADY-OPEN</td>
<td>Element is already open.</td>
</tr>
<tr>
<td>ERROPERNOTINCTX</td>
<td>Element is not in the current context.</td>
</tr>
<tr>
<td>NOCONTEXT</td>
<td>There is no current context.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_TO_NONE</td>
<td>Promote only this element. This is the default.</td>
</tr>
<tr>
<td>MCS_TO_BOTTOM</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.

_________________________ 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.
promote

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISRESERVED</td>
<td>element is reserved.</td>
</tr>
<tr>
<td>NEEDCONTROL</td>
<td>element is uncontrolled.</td>
</tr>
<tr>
<td>NOBASEPART</td>
<td>The current context has no base partition set.</td>
</tr>
<tr>
<td>NOCONTEXT</td>
<td>There is no currently opened context.</td>
</tr>
<tr>
<td>NOTVERSIONED</td>
<td>element is not a versioned object.</td>
</tr>
<tr>
<td>NOTVISIBLE</td>
<td>element is not visible in the current context.</td>
</tr>
<tr>
<td>NOTMOD</td>
<td>element cannot be modified in the current context.</td>
</tr>
<tr>
<td>PROFOROROO</td>
<td>element is in the root partition and cannot be promoted.</td>
</tr>
</tbody>
</table>
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  
6 TEXT_tool inherits  
4 COMPOSITE inherits  
5 COLLECTION inherits  
3 MESSAGE inherits
purge

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
rename

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_TO_NONE</td>
<td>Replace only this element. This is the default.</td>
</tr>
</tbody>
</table>
replace

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_TO TOP</td>
<td>Replace this element and all direct and indirect parents to the top of the composite hierarchy, provided that they do not own 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.</td>
</tr>
<tr>
<td>MCS_TO BOTTOM</td>
<td>Replace this element and all elements owned, either directly or indirectly, through dependency relationships. These include direct and indirect children of a composite.</td>
</tr>
<tr>
<td>MCS_TO BOTH</td>
<td>Equivalent to sending replace twice, once with MCS_TO_BOTTOM specified, then with MCS_TO_TOP.</td>
</tr>
</tbody>
</table>

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
  6 TEXT.Tool inherits
4 COMPOSITE inherits
  5 COLLECTION refines
3 MESSAGE refines
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.

Message Descriptions  2–65
replace


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—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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_TO_NONE</td>
<td>Reserve only this element. This is the default.</td>
</tr>
<tr>
<td>MCS_TO_TOP</td>
<td>Reserve this element and all direct and indirect parents to the top of the composite hierarchy.</td>
</tr>
<tr>
<td>MCS_TO_BOTTOM</td>
<td>Reserve this element and all direct and indirect children in the composite hierarchy.</td>
</tr>
<tr>
<td>MCS_TO_BOTH</td>
<td>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.</td>
</tr>
<tr>
<td>MCS_TO_CLOSURE</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.
reserve

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.

Message Descriptions 2–71
reserve

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

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

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.
setProp

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_COMPUTEDPROPERTY inherits
    4 HAS_RELATION_PROPERTY inherits
 3 HAS_RELATION inherits
 3 HAS_SUPERTYPE inherits
 3 IMPLEMENTS_METHOD inherits
setProp

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 HASRELATED_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.

For top:
1. Verify that element owns no reserved elements or open files.
2. Recursively delete all directories under the context directory.

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.
ELEMENT
1. Verify that \texttt{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 \texttt{setProp} message must have been dispatched by a \texttt{new} method
3. Set the value of the property to its new value.
4. For properties other than \texttt{history}, send a notification to each owner of a dependence relationship for which \texttt{element} is a member.

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

PARTITION
For \texttt{partitionDir}:
1. Verify that there are no open files in the current partition.
2. Verify that the supplied value is not null.
4. Rename \texttt{element}'s partition directory to the new directory name.

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

**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
6 TEXT inherits
7 TEXT_TOOL inherits
4 COMPOSITE inherits
5 COLLECTION inherits
3 MESSAGE inherits
3 MSGARG inherits
3 TOOL inherits
translate

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
6 TEXT_tool inherits
7 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
unfreeze

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_TO_NONE</td>
<td>Unreserve only this element. This is the default.</td>
</tr>
<tr>
<td>MCS_TO_TOP</td>
<td>Unreserve this element and all direct and indirect parents to the top of the composite hierarchy.</td>
</tr>
<tr>
<td>MCS_TO_BOTTOM</td>
<td>Unreserve this element and all direct and indirect children in the composite hierarchy.</td>
</tr>
</tbody>
</table>
unreserve

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_TO_BOTH</td>
<td>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.</td>
</tr>
</tbody>
</table>

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
6 TEXT inherits
7 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.
unreserve

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
Disallowled. Metadata elements cannot be unreserved.

METHOD
Disallowled. Metadata elements cannot be unreserved.

TYPE
Disallowled. 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.
unreserve

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.
OWNRESERVED 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.
update

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.
verify

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_VERIFY_INTERNAL_STRUCTURE</td>
<td>Check internal structure.</td>
</tr>
<tr>
<td>MCS_VERIFY_REPAIR</td>
<td>Repair the object.</td>
</tr>
</tbody>
</table>

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

2–90 Message Descriptions
verify

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 HASPROPERTY 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.
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—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
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.
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

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element Type Privileges</strong></td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td>0</td>
</tr>
<tr>
<td>WRITE</td>
<td>1</td>
</tr>
<tr>
<td>MODIFY</td>
<td>2</td>
</tr>
<tr>
<td>ERASE</td>
<td>3</td>
</tr>
<tr>
<td>NOREAD</td>
<td>16</td>
</tr>
<tr>
<td>NOWRITE</td>
<td>17</td>
</tr>
<tr>
<td>NOMODIFY</td>
<td>18</td>
</tr>
<tr>
<td>NOERASE</td>
<td>19</td>
</tr>
</tbody>
</table>

| **Element Instance Privileges** |     |
| SHOW          | 4   |
| DEFINE        | 5   |
| CHANGE        | 6   |
| DELETE        | 7   |
| CONTROL       | 8   |
| NOSHOW        | 20  |
| NODEFINDE     | 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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_PROP_ACCESS_READONLY</td>
<td>Read-only</td>
</tr>
<tr>
<td>MCS_PROP_ACCESS_READWRITE</td>
<td>Read/write</td>
</tr>
<tr>
<td>MCS_PROP_ACCESS_WRITEONCE</td>
<td>Write-once</td>
</tr>
<tr>
<td>MCS_PROP_ACCESS_WRITECREATE</td>
<td>Write-once-at-creation</td>
</tr>
</tbody>
</table>

aliases—Command Aliases

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

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>MCS_prop_aliases</th>
<th>MCS$r_prop_aliases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td></td>
<td></td>
<td>NAD$K_ATT_ALIASES</td>
<td></td>
</tr>
<tr>
<td>Defined By</td>
<td></td>
<td></td>
<td>PERSISTENT_PROCESS</td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td></td>
<td></td>
<td>MCS_LIST</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td></td>
<td></td>
<td>Read/write</td>
<td></td>
</tr>
</tbody>
</table>

allCheckouts—All Reserved Versions

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

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>MCS_prop_allCheckouts</th>
<th>MCS$r_prop_allCheckouts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined By</td>
<td>DATABASE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
<td>Computed</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td></td>
<td></td>
<td>MCS_SCAN</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td></td>
<td></td>
<td>Read-only</td>
<td></td>
</tr>
</tbody>
</table>
allChildPartitions

**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined By</td>
<td>PARTITION</td>
<td>No</td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td>Closure</td>
</tr>
<tr>
<td>Type</td>
<td>Closure</td>
<td>Read-only</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_SCAN</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relation Traversed</td>
<td>HAS_PARENT</td>
<td></td>
</tr>
<tr>
<td>Traversal Direction</td>
<td>To all owners</td>
<td></td>
</tr>
<tr>
<td>Inverse Property</td>
<td>allParentPartitions</td>
<td></td>
</tr>
</tbody>
</table>

**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
OpenVMS binding
Defined By
Required With new
Type
Data Type
Access
Relation Traversed
Traversal Direction
Inverse Property

MCS_prop_allChildren
MCS$r_prop_allChildren
COMPOSITE
No
Closure
MCS_SCAN
Read-only
COMPOSITE_PART
To all members
None

allDependencies—All Dependencies

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

Summary

Symbolic Name
C binding
OpenVMS binding
Defined By
Required With new
Type
Data Type
Access
Relation Traversed
Traversal Direction
Inverse Property

MCS_prop_allDependencies
MCS$r_prop_allDependencies
VERSION
No
Closure
MCS_SCAN
Read-only
DEPENDS_ON
To all owners
allDependents
allDependents—All Dependents

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

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>MCS_prop_allDependents</th>
<th>MCS$r_prop_allDependents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined By</td>
<td>VERSION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Closure</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_SCAN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relation Traversed</td>
<td>DEPENDS_ON</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traversal Direction</td>
<td>To all members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inverse Property</td>
<td>allDependencies</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>C binding</td>
<td>VERSION</td>
<td>new</td>
<td>Closure</td>
<td>MCS_SCAN</td>
<td>Read-only</td>
<td>METHOD_PARAMETER</td>
<td>To all members</td>
<td>allDerives</td>
</tr>
<tr>
<td>OpenVMS binding</td>
<td></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>C binding</td>
<td>VERSION</td>
<td>new</td>
<td>Closure</td>
<td>MCS_SCAN</td>
<td>Read-only</td>
<td>METHOD_PARAMETER</td>
<td>To all owners</td>
<td>allDerivedFrom</td>
</tr>
<tr>
<td>OpenVMS binding</td>
<td></td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.
allSubTypes

Summary

Symbolic Name
  C binding
  OpenVMS binding

Defined By
Type

Required With new
No

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
  OpenVMS binding

Tag

Defined By
TYPE

Required With new
No

Type
Closure

Data Type
MCS_SCAN

Access
Read-only

Relation Traversed
HASSUPERTYPE

Traversal Direction
To all members

Inverse Property
allSubTypes
alternateNames

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
OpenVMS binding

MCS_prop_alternateNames
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
OpenVMS binding

MCS_prop_application
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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_argSpec</td>
<td>MCS$r_prop_argSpec</td>
<td>MESSAGE</td>
<td>No</td>
<td>Relation</td>
<td>MCS_SCAN</td>
<td>Read/write</td>
<td>HAS_MSGARG</td>
<td>To member</td>
<td>messagesHavingMsgarg</td>
</tr>
</tbody>
</table>

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

associatedValidations—Associated Validations

Identifies VALIDATION elements that operate on an element type.

Summary

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_ATTACH_SPEC_VERSION</td>
<td>Attach specific version.</td>
</tr>
<tr>
<td>MCS_ATTACH_LAST_CHKIN</td>
<td>Attach last checked-in version.</td>
</tr>
<tr>
<td>MCS_ATTACH_LATEST</td>
<td>Attach reserved or last checked-in version.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_attachmentInContext</td>
<td>MCS$r_prop_attachmentInContext</td>
</tr>
</tbody>
</table>

Defined By: VERSION
Required With: No
Type: Computed
Data Type: MCS_LIST
Access: Read-only

Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_ATTACH_SPEC_VERSION</td>
<td>Attach specific version.</td>
</tr>
<tr>
<td>MCS_ATTACH_LAST_CHKIN</td>
<td>Always use last checked in version.</td>
</tr>
<tr>
<td>MCS_ATTACH_LATEST</td>
<td>Reserved or last checked in version.</td>
</tr>
</tbody>
</table>

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.
autopurge

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Intermediate versions should be purged.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Intermediate versions should not be purged.</td>
</tr>
</tbody>
</table>

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
OpenVMS binding
Defined By
Required With new
Type
Data Type
Access

MCS_prop_availVersion
MCS$r_prop_availVersion
VERSION
No
Computed
MCS_ELEMENTID
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
OpenVMS binding
Defined By
Required With new
Type
Data Type
Access
Relation Traversed
Traversal Direction
Inverse Property

MCS_prop_basePartition
MCS$r_prop_basePartition
CONTEXT
Yes
Relation
MCS_ELEMENTID
Write-once-at-creation
HAS_PARENT
To member
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**

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

---

**baseTypeSize—Base Data Type Size**

Indicates the size of the base type in bytes.

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>new</td>
<td>Normal</td>
<td>MCS_SMALLINT</td>
<td>Write-once-at-creation</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>new</td>
<td>Normal</td>
<td>MCS_STRING</td>
<td>Read-only</td>
</tr>
</tbody>
</table>
checkout—Reserved Versions

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

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>MCS_prop_checkout</th>
</tr>
</thead>
<tbody>
<tr>
<td>C binding</td>
<td>MCS$r_prop_checkout</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defined By</th>
<th>CONTEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required With new</td>
<td>No</td>
</tr>
<tr>
<td>Type</td>
<td>Relation</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_SCAN</td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
</tr>
<tr>
<td>Relation Traversed</td>
<td>RESERVED_BY</td>
</tr>
<tr>
<td>Traversal Direction</td>
<td>To owner</td>
</tr>
<tr>
<td>Inverse Property</td>
<td>reservedBy</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_childPartitions</td>
<td>MCS$r_prop_childPartitions</td>
<td>PARTITION</td>
<td>No</td>
<td>Relation</td>
<td>MCS_SCAN</td>
<td>Read-only</td>
<td>HAS_PARENT</td>
<td>To owner</td>
<td>parentPartition</td>
</tr>
</tbody>
</table>

### 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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_compPropDef</td>
<td>MCS$r_prop_compPropDef</td>
<td>ELEMENT_TYPE</td>
<td>No</td>
<td>Relation</td>
<td>MCS_SCAN</td>
<td>Read/write</td>
<td>HAS_COMPUTED_PROPERTY</td>
<td>To member</td>
<td>typesHavingCompProp</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextDir</td>
<td>MCS_prop_contextDir</td>
<td>MCS$r_prop_contextDir</td>
<td>NAD$K_ATT_CONTEXTDIR</td>
<td>CONTEXT</td>
<td>No</td>
<td>Normal</td>
<td>MCS_STRING</td>
<td>Read/write</td>
</tr>
</tbody>
</table>

contextName—Context Name

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

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>contextName</td>
<td>MCS_prop_contextName</td>
<td>MCS$r_prop_contextName</td>
<td>NAD$K_ATT_HISTORY_CTX_NAME</td>
<td>EVENT</td>
<td>No</td>
<td>Normal</td>
<td>MCS_STRING</td>
<td>Read/write</td>
</tr>
</tbody>
</table>
**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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>MCS_prop_controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td>C binding</td>
<td>MCS$r_prop_controlled</td>
</tr>
<tr>
<td>OpenVMS binding</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>NAD$K_ATT_CONTROLLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined By</td>
<td>VERSION</td>
</tr>
<tr>
<td>Required With <strong>new</strong></td>
<td>No</td>
</tr>
<tr>
<td>Type</td>
<td>Normal</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_BOOLEAN</td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
</tr>
</tbody>
</table>

### Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Version is controlled.</td>
</tr>
<tr>
<td>FALSE</td>
<td>Version is not controlled.</td>
</tr>
</tbody>
</table>

**CPUTime—CPU Time Used**

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

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_CPUTime</td>
<td>MCS$r_prop_CPUTime</td>
<td>METHOD_INVOCATION</td>
<td>No</td>
<td>Normal</td>
<td>MCS_LONGINT</td>
<td>Write-once-at-creation</td>
</tr>
</tbody>
</table>

### createdDate—Creation Date and Time

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

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_createdDate</td>
<td>MCS$r_prop_createdDate</td>
<td>EVENT METHOD_INVOCATION NAMED_ELEMENT</td>
<td>No</td>
<td>Normal</td>
<td>MCS_DATETIME</td>
<td>Write-once-at-creation</td>
</tr>
</tbody>
</table>
**currCollection**

---

**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.
currContext

Summary

Symbolic Name
C binding
OpenVMS binding
MCS_prop_currContext
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
OpenVMS binding
MCS_prop_dataType
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

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
OpenVMS binding
Tag
Defined By
Required With new
Type
Data Type
Access
MCS_prop_defaultAccess
MCS$r_prop_defaultAccess
NAD$K_ATT_DEFAULT_ACL
CONTEXT
DATABASE
PERSISTENT_PROCESS
No
Normal
MCS_MEMBLOCK
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
OpenVMS binding
Tag
Defined By
Required With new
Type
Data Type
Access

MCS_prop_defaultAttachment
MCS$r_prop_defaultAttachment
NAD$K_ATT_DEFAULT_ATTACHMENT
CONTEXT
No
Normal
MCS_SMALLINT
Read/write

Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_ATTACH_SPEC_VERSION</td>
<td>Attach specific version.</td>
</tr>
<tr>
<td>MCS_ATTACH_LAST_CHKIN</td>
<td>Attach last checked-in version.</td>
</tr>
<tr>
<td>MCS_ATTACH_LATEST</td>
<td>Attach reserved or last checked-in version.</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With <strong>new</strong></th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_definedMethods</td>
<td>MCS$r_prop_definedMethods</td>
<td>ELEMENT_TYPE</td>
<td>No</td>
<td>Computed</td>
<td>MCS_SCAN</td>
<td>Read-only</td>
</tr>
</tbody>
</table>

### 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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With <strong>new</strong></th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_definedPropDefs</td>
<td>MCS$r_prop_definedPropDefs</td>
<td>ELEMENT_TYPE</td>
<td>No</td>
<td>Computed</td>
<td>MCS_SCAN</td>
<td>Read-only</td>
</tr>
</tbody>
</table>
deltaFile

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
  OpenVMS binding
Tag
Defined By
Required With new
Type
Data Type
Access

dependencies—Dependencies

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

Summary

Symbolic Name
C binding
OpenVMS binding

MCS_prop_dependencies
MCS$r_prop_dependencies

Defined By
Required With new
Type
Data Type
Access
Relation Traversed
Traversal Direction
Inverse Property

VERSION
No
Relation
MCS_SCAN
Read/write
DEPENDS_ON
To owner
dependents

dependents—Dependents

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

Summary

Symbolic Name
C binding
OpenVMS binding

MCS_prop_dependents
MCS$r_prop_dependents

Defined By
Required With new
Type
Data Type
Access
Relation Traversed
Traversal Direction
Inverse Property

VERSION
No
Relation
MCS_SCAN
Read/write
DEPENDS_ON
To member
dependencies
derivedFrom

---

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

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.
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_RELATIONPROPERTY

Required With: new

Type: Normal

Data Type: MCS_SMALLINT

Access: Read/write

Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_DIRECTION_TO_MEMBER</td>
<td>Traverse the relation from owner to member.</td>
</tr>
<tr>
<td>MCS_DIRECTION_TO_ALL_MEMBERS</td>
<td>Traverse the relation from owner to all members.</td>
</tr>
<tr>
<td>MCS_DIRECTION_TO_OWNER</td>
<td>Traverse the relation from member to owner.</td>
</tr>
<tr>
<td>MCS_DIRECTION_TO_ALL_OWNERS</td>
<td>Traverse the relation from member to all owners.</td>
</tr>
</tbody>
</table>


**elapsedTime—Elapsed Time**

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

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>MCS_prop_elapsedTime</th>
<th>MCS$r_prop_elapsedTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>NAD$K_ATT_ELAPSEDTIME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defined By</td>
<td>METHOD_INVOCATION</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Normal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_LONGINT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Write-once-at-creation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>MCS_prop_elementType</th>
<th>MCS$r_prop_elementType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined By</td>
<td>ELEMENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Computed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_ELEMENTID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
filePath

**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_filePath</td>
<td>MCS$r_prop_filePath</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_prop_firstVersion</td>
<td>MCS$r_prop_firstVersion</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defined By</th>
<th>VERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required With new</td>
<td>No</td>
</tr>
<tr>
<td>Type</td>
<td>Computed</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_ELEMENTID</td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
</tr>
</tbody>
</table>

freezeTime—Replacement Time

Indicates the time at which a VERSION element was replaced.

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_prop_freezeTime</td>
<td>MCS$r_prop_freezeTime</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>NAD$K_ATT_FREEZE_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined By</td>
<td>VERSION</td>
</tr>
<tr>
<td>Required With new</td>
<td>No</td>
</tr>
<tr>
<td>Type</td>
<td>Normal</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_DATETIME</td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
</tr>
</tbody>
</table>
funcType

**funcType—Method Function Type**

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

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_funcType</td>
<td>MCS$r_prop_funcType</td>
<td>NAD$K_ATT_FUNCTYPE</td>
<td>METHOD</td>
<td>Yes</td>
<td>Normal</td>
<td>MCS_SMALLINT</td>
<td>Read/write</td>
</tr>
</tbody>
</table>

**Constant Values**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_METHOD_INTERNAL_SCRIPT</td>
<td>Invokes an internal script. This method type is not supported in this release of Oracle CDD/Repository.</td>
</tr>
<tr>
<td>MCS_METHOD_NULL</td>
<td>Invokes a null function.</td>
</tr>
<tr>
<td>MCS_METHOD_TRANSPARENT</td>
<td>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.</td>
</tr>
<tr>
<td>MCS_METHOD_SUPEROP</td>
<td>Invokes the supertype's method for this message. This function type provides a convenient means of creating a placeholder for preambles and postambles.</td>
</tr>
<tr>
<td>MCS_METHOD_ILLEGAL</td>
<td>Specifies an illegal function. Invoking the method results in an error indicating that no method was specified for the message.</td>
</tr>
</tbody>
</table>

###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.
hasChildren

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_hasChildren</td>
<td>MCS$r_prop_hasChildren</td>
</tr>
<tr>
<td>Defined By</td>
<td>COMPOSITE</td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td>No</td>
</tr>
<tr>
<td>Type</td>
<td>Relation</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_SCAN</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
<td></td>
</tr>
<tr>
<td>Relation Traversed</td>
<td>COMPOSITE_PART</td>
<td></td>
</tr>
<tr>
<td>Traversal Direction</td>
<td>To member</td>
<td></td>
</tr>
<tr>
<td>Inverse Property</td>
<td>hasParents</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_hasParents</td>
<td>MCS$r_prop_hasParents</td>
</tr>
<tr>
<td>Defined By</td>
<td>VERSION</td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td>No</td>
</tr>
<tr>
<td>Type</td>
<td>Relation</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_SCAN</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
<td></td>
</tr>
<tr>
<td>Relation Traversed</td>
<td>COMPOSITE_PART</td>
<td></td>
</tr>
<tr>
<td>Traversal Direction</td>
<td>To owner</td>
<td></td>
</tr>
<tr>
<td>Inverse Property</td>
<td>hasChildren</td>
<td></td>
</tr>
</tbody>
</table>
**history—History**

Identifies all the `EVENT` elements for an element.

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MCS_prop_history</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MCS$r_prop_history</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NAMED_ELEMENT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>Relation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**historyComment—History Comment**

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

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EVENT</td>
<td>No</td>
<td>Normal</td>
<td>MCS_LIST</td>
<td>Write-once-at-creation</td>
</tr>
</tbody>
</table>
implementsMessage

**implementsMessage—Message Implemented by Method**

Identifies the message for which this method is the implementation.

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop ImplementsMessage</td>
<td>MCS$r_prop ImplementsMessage</td>
<td>METHOD</td>
<td>No</td>
<td>Relation</td>
<td>MCS_ELEMENTID</td>
<td>Read/write</td>
<td>HAS_MESSAGE</td>
<td>To member</td>
<td>implementingMethods</td>
</tr>
</tbody>
</table>

**implementsProperty—Method Implementing Computed Property**

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

Summary

Symbolic Name
- C binding: MCS_prop_implementsMethod
- OpenVMS binding: MCS$r_prop_implementsMethod

Defined By: HAS_COMPUTED_PROPERTY

Required With: new

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

Type: Relation

Data Type: MCS_ELEMENTID

Access: Read/write

Relation Traversed: IMPLEMENTS_RELATION

Traversal Direction: To member

Inverse Property: relPropUsing
importedFrom

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 \texttt{ELEMENT\_TYPE} and its subtypes, identifies all instances of the type. (Compare with \texttt{allInstances}, which includes instances of subtypes.)

For \texttt{PARTITION}, identifies the versions that reside in a partition.

Summary

\begin{tabular}{ll}
Symbolic Name & \texttt{MCS\_prop\_instances} \\
\texttt{C binding} & \texttt{MCS\_r\_prop\_instances} \\
\texttt{OpenVMS binding} & \\
Defined By & \texttt{ELEMENT\_TYPE} \\
& \texttt{PARTITION} \\
Required With \texttt{new} & No \\
Type & Computed \\
Data Type & \texttt{MCS\_SCAN} \\
Access & Read-only \\
\end{tabular}

\hspace{1cm}

invocationStatus—Method Invocation Status

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

Summary

\begin{tabular}{ll}
Symbolic Name & \texttt{MCS\_prop\_invocationStatus} \\
\texttt{C binding} & \texttt{MCS\_r\_prop\_invocationStatus} \\
\texttt{OpenVMS binding} & \\
Tag & \texttt{NAD\$K\_ATT\_INVOCATIONSTATUS} \\
Defined By & \texttt{METHOD\_INVOCATION} \\
Required With \texttt{new} & No \\
Type & Normal \\
Data Type & \texttt{MCS\_LONGINT} \\
Access & Write-once-at-creation \\
\end{tabular}
invocationString

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TOOL</td>
<td></td>
<td></td>
<td>INVOICES_TOOL</td>
<td>To member</td>
<td>invokedBy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MCS_ELEMENTID</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>NAD$K_ATT_KEEPHIST</td>
<td>METHOD</td>
<td></td>
<td>Normal</td>
<td>MCS_BOOLEAN</td>
<td>Read/write</td>
</tr>
</tbody>
</table>

Property Descriptions 3–51
keepHist

## Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>A history record will be created each time the method is executed.</td>
</tr>
<tr>
<td>FALSE</td>
<td>A history record will not be created each time the method is executed.</td>
</tr>
</tbody>
</table>

## 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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>MCS_prop_lastVersion</th>
</tr>
</thead>
<tbody>
<tr>
<td>C binding</td>
<td>MCS$r_prop_lastVersion</td>
</tr>
<tr>
<td>OpenVMS binding</td>
<td></td>
</tr>
<tr>
<td>Defined By</td>
<td>VERSION</td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
</tr>
<tr>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Computed</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_ELEMENTID</td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
</tr>
</tbody>
</table>
legalMembers

---

**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_legalMembers</td>
<td>MCS$r_prop_legalMembers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defined By</th>
<th>RELATION_TYPE</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Required With</th>
<th>new</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Relation</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_SCAN</td>
</tr>
<tr>
<td>Access</td>
<td>Read/write</td>
</tr>
<tr>
<td>Relation Traversed</td>
<td>RELATION_MEMBER</td>
</tr>
<tr>
<td>Traversal Direction</td>
<td>To member</td>
</tr>
<tr>
<td>Inverse Property</td>
<td>relationMember</td>
</tr>
</tbody>
</table>

---

**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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>RELATION_TYPE</td>
<td>new</td>
<td>Relation</td>
<td>MCS_SCAN</td>
<td>Read/write</td>
<td>HAS_RELATION</td>
<td>To owner</td>
<td>ownsRelation</td>
</tr>
</tbody>
</table>

**logFile—Log File**

Names a file that contains the execution log of the tool invoked in this method invocation.

### Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>METHOD_INVOCATION</td>
<td>new</td>
<td>Relation</td>
<td>MCS_STRING</td>
<td>Write-once-at-creation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

3–54 Property Descriptions
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

---

**methodType—Method Type**

Identifies the element type for which a method implements a message.

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_methodType</td>
<td>MCS$r_prop_methodType</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defined By</th>
<th>METHOD</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Required With</th>
<th>new</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>MCS_ELEMENTID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Read-only</td>
</tr>
</tbody>
</table>

methodUsed—Method Used

Identifies the METHOD element that was invoked in this method invocation.

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_methodUsed</td>
<td>MCS$r_prop_methodUsed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defined By</th>
<th>ATIS_METHOD_INVOC</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Required With</th>
<th>new</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Relation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>MCS_ELEMENTID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Write-once-at-creation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relation Traversed</th>
<th>HAS_METHOD_USED</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Traversal Direction</th>
<th>To member</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Inverse Property</th>
<th>miUsingMethod</th>
</tr>
</thead>
</table>
msgSent—Message Sent

Identifies the MESSAGE element that was sent to start this method invocation.

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_msgSent</td>
<td>MCS$r_prop_msgSent</td>
<td>ATIS_METHOD_INVOC</td>
<td>No</td>
<td>Relation</td>
<td>MCS_ELEMENTID</td>
<td>Write-once-at-creation</td>
<td>HAS_MSG_SENT</td>
<td>To member</td>
<td>miForMessage</td>
</tr>
</tbody>
</table>

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
OpenVMS binding
Defined By
Required With new
Type
Data Type
Access
Relation Traversed
Traversal Direction
Inverse Property

C binding
OpenVMS binding
ATIS_METHOD_INVOC
No
Relation
MCS_ELEMENTID
Write-once-at-creation
HAS_MSG_TARGET
To member
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
OpenVMS binding
Tag
Defined By
Required With new
Type
Data Type
Access

MCS_propMutable
MCS$r_propMutable
NAD$K_ATT_MUTABLE
HAS_PROPERTY
No
Normal
MCS_BOOLEAN
Read/write
mutable

Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The property can be changed even if the owning element is replaced.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The property cannot be changed when the owning element is replaced.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With new</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>NAD$K_ATT_NAD_MCSNAME</td>
<td>NAMED_ELEMENT</td>
<td>Yes, for the following subtypes of NAMED_ELEMENT: BINARY, COLLECTION, CONTEXT, DIRECTORY, MESSAGE, METHOD, MSGARG, PARTITION, PERSISTENT_PROCESS, and TYPE</td>
<td>Normal</td>
<td>MCS_STRING</td>
<td>Read/write</td>
</tr>
</tbody>
</table>

Property Descriptions 3-59
nextVersions

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 | MCS_prop_nextVersions | MCS$r_prop_nextVersions
---|---|---
C binding | | |
OpenVMS binding | | |
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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>MCS_prop_node</th>
<th>C binding</th>
<th>MCS$r_prop_node</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenVMS binding</td>
<td></td>
<td></td>
<td>OpenVMS binding</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td></td>
<td>NAD$K_ATT_NODE_NAME</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defined By</td>
<td>DATABASE</td>
<td></td>
<td>Defined By</td>
<td>DATABASE</td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td>Yes</td>
<td>Required With</td>
<td>new</td>
</tr>
<tr>
<td>Type</td>
<td>Normal</td>
<td></td>
<td>Type</td>
<td>Normal</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_STRING</td>
<td></td>
<td>Data Type</td>
<td>MCS_LONGINT</td>
</tr>
<tr>
<td>Access</td>
<td>Read/write</td>
<td></td>
<td>Access</td>
<td>Read-only</td>
</tr>
</tbody>
</table>

### numChildren—Number of Composite Children

Indicates the number of versions attached to a composite.

### Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>MCS_prop_numChildren</th>
<th>C binding</th>
<th>MCS$r_prop_numChildren</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenVMS binding</td>
<td></td>
<td></td>
<td>OpenVMS binding</td>
<td></td>
</tr>
<tr>
<td>Defined By</td>
<td>COMPOSITE</td>
<td></td>
<td>Defined By</td>
<td>COMPOSITE</td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td>No</td>
<td>Required With</td>
<td>new</td>
</tr>
<tr>
<td>Type</td>
<td>Computed</td>
<td></td>
<td>Type</td>
<td>Computed</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_LONGINT</td>
<td></td>
<td>Data Type</td>
<td>MCS_LONGINT</td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
<td></td>
<td>Access</td>
<td>Read-only</td>
</tr>
</tbody>
</table>
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.
openedFiles

Summary

Symbolic Name
C binding MEMBERS
OpenVMS binding MEMBERS
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 MEMBERS
OpenVMS binding MEMBERS
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

OSVersion—Operating System Version

Indicates the operating system version number under which this method invocation occurred.

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_prop_OSVersion</td>
<td>MCS$r_prop_OSVersion</td>
<td></td>
</tr>
<tr>
<td>Tag</td>
<td>NAD$K_ATT_OSVERSION</td>
<td></td>
</tr>
<tr>
<td>Defined By</td>
<td>METHOD_INVOCATION</td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_STRING</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Write-once-at-creation</td>
<td></td>
</tr>
</tbody>
</table>

owner—Element Owner

Contains a system-specific identifier for the user who created the element.

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_prop_owner</td>
<td>MCS$r_prop_owner</td>
<td></td>
</tr>
<tr>
<td>Defined By</td>
<td>NAMED_ELEMENT</td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Computed</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>System-specific identifier</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_prop_ownsRelation</td>
<td></td>
<td>MCS$r_prop_ownsRelation</td>
<td>ELEMENT_TYPE</td>
<td>No</td>
<td>Relation</td>
<td>MCS_SCAN</td>
<td>Read/write</td>
<td>HAS_RELATION</td>
<td>To member</td>
<td>legalOwners</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_prop_parentInContext</td>
<td></td>
<td>MCS$r_prop_parentInContext</td>
<td>VERSION</td>
<td>No</td>
<td>Computed</td>
<td>MCS_SCAN</td>
<td>Read-only</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
parentPartition

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.
### partitionDir

#### 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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_MSGARG_IN</td>
<td>Input argument, read-only.</td>
</tr>
<tr>
<td>MCS_MSGARG_OUT</td>
<td>Output argument, write-only.</td>
</tr>
<tr>
<td>MCS_MSGARG_INOUT</td>
<td>Input and output argument, read/write.</td>
</tr>
</tbody>
</table>
### path—Repository Path

Contains the native file system path name of the repository, or its name service handle if a name service is available.

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With new</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_path</td>
<td>MCS$r_prop_path</td>
<td>NAD$K_ATT_ANCHOR_NAME</td>
<td>DATABASE</td>
<td>Yes</td>
<td>Normal</td>
<td>MCS_STRING</td>
<td>Read/write</td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With new</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_pattern</td>
<td>MCS$r_prop_pattern</td>
<td>NAD$K_ATT_PATTERN</td>
<td>ELEMENT_TYPE</td>
<td>No</td>
<td>Normal</td>
<td>MCS_STRING</td>
<td>Read/write</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_postamble</td>
<td>MCS$r_prop_postamble</td>
</tr>
</tbody>
</table>

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.
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.
**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_processingName</td>
<td>MCS$r_prop_processingName</td>
<td>NAD$K_ATT_PROCESSING_NAME</td>
<td>ELEMENT</td>
<td>No</td>
<td>Normal</td>
<td>MCS_STRING</td>
<td>Read/write</td>
</tr>
</tbody>
</table>

**propDef—Properties**

Identifies all properties defined for the type and for its supertypes.
propDef

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_propDef</td>
<td>MCS$r_prop_propDef</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_referenceCount</td>
<td>MCS$r_prop_referenceCount</td>
</tr>
</tbody>
</table>

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
OpenVMS binding
Defined By
Required With new
Type
Data Type
Access
Relation Traversed
Traversal Direction
Inverse Property

MCS_prop_relationMember
MCS$r_prop_relationMember
ELEMENT_TYPE
No
Relation
MCS_SCAN
Read/write
RELATION_MEMBER
To owner
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
OpenVMS binding
Tag
Defined By
Required With new
Type
Data Type
Access

MCS_prop_relMember
MCS$r_prop_relMember
NAD$K_ATT_REL_MEMBER
RELATION
Yes
Normal
MCS_SCAN
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
OpenVMS binding
Tag
Defined By
Required With
Type
Data Type
Access

MCS_prop_relOwner
MCS$r_prop_relOwner
NAD$K_ATT_REL_OWNER
RELATION
new
Yes
Normal
MCS_SCAN
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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>ELEMENT_TYPE</td>
<td>new</td>
<td>Relation</td>
<td>MCS_SCAN</td>
<td>Read/write</td>
<td>HAS_RELATIONPROPERTY</td>
<td>To member</td>
<td>typesHavingRelProp</td>
</tr>
</tbody>
</table>

#### 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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>NAD$K_ATT_REQUIRED</td>
<td>HAS_MSGARG</td>
<td>new</td>
<td>Normal</td>
<td>MCS_SMALLINT</td>
<td>Read/write</td>
<td>typesHavingRelProp</td>
</tr>
</tbody>
</table>
Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>A value is required.</td>
</tr>
<tr>
<td>FALSE</td>
<td>A value is not required.</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_rootBranchInstances</td>
<td>MCS$r_prop_rootBranchInstances</td>
</tr>
<tr>
<td>Defined By</td>
<td>ELEMEN_T_TYPE</td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td>No</td>
</tr>
<tr>
<td>Type</td>
<td>Computed</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_SCAN</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
<td></td>
</tr>
</tbody>
</table>

**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_ATTANCHORROOT_NAME
Defined By  DATABASE
Required With  new No
Type  Normal
Data Type  MCS_STRING
Access  Write-once
rootVersion

**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

**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_scalingFactor</td>
<td>MCS$r_prop_scalingFactor</td>
</tr>
<tr>
<td>Tag</td>
<td>NAD$K_ATT_SCALINGFACTOR</td>
<td></td>
</tr>
<tr>
<td>Defined By</td>
<td>METHOD_INVOCATION</td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_FLOAT</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Write-once-at-creation</td>
<td></td>
</tr>
</tbody>
</table>

**simpleName—Simple Name**

Contains the name of an element without any repository or directory information at the front.

**Summary**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_simpleName</td>
<td>MCS$r_prop_simpleName</td>
</tr>
<tr>
<td>Defined By</td>
<td>NAMED_ELEMENT</td>
<td></td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Computed</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_STRING</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Read-only</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_STS_AVAIL</td>
<td>Available for reserve—The version is last version on the line of descent and is replaced.</td>
</tr>
<tr>
<td>MCS_STS_RO</td>
<td>Read-only—The version has descendants on the same branch, and may be reserved only if a new branch is created.</td>
</tr>
<tr>
<td>MCS_STS_GHOST</td>
<td>The version is reserved.</td>
</tr>
<tr>
<td>MCS_STS_FROZEN</td>
<td>The version may not be reserved.</td>
</tr>
</tbody>
</table>

The values of the status property are mutually exclusive.
storedIn

---

**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**

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_storedIn</td>
<td>MCS$r_prop_storedIn</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>NAD$K_ATT_STOREDIN</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Defined By</th>
<th>BINARY</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Required With</th>
<th>new</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes (if the value of <code>storeType</code> is MCS_STORETYPE_EXTERNAL)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Normal</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Data Type</th>
<th>MCS_STRING</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Access</th>
<th>Read/write</th>
</tr>
</thead>
</table>

**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
Type: Normal
Data Type: MCS_SMALLINT
Access: Write-once-at-creation

Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS_STORETYPE_EXTERNAL</td>
<td>File is under Oracle CDD/Repository control.</td>
</tr>
<tr>
<td>MCS_STORETYPE_INTERNAL</td>
<td>File is not under Oracle CDD/Repository control.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With new</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_subTypes</td>
<td>MCS$r_prop_subTypes</td>
<td>TYPE</td>
<td>No</td>
<td>Relation</td>
<td>MCS_SCAN</td>
<td>Read-only</td>
<td>HAS_SUPER_TYPE</td>
<td>To owner</td>
<td>superTypes</td>
</tr>
</tbody>
</table>

---

### 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.

_________________________
**superTypes**

### Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
<th>Relation Traversed</th>
<th>Traversal Direction</th>
<th>Inverse Property</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_superTypes</td>
<td>MCS$r_prop_superTypes</td>
<td>TYPE</td>
<td>Yes (required by ELEMENT_TYPE)</td>
<td>Relation</td>
<td>MCS_SCAN</td>
<td>Write-once-at-creation</td>
<td>HAS_SUPER_TYPE</td>
<td>To member</td>
<td>subTypes</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
<th>Tag</th>
<th>Defined By</th>
<th>Required With</th>
<th>Type</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_symbols</td>
<td>MCS$r_prop_symbols</td>
<td>NAD$K_ATT_SYMBOLS</td>
<td>PERSISTENT_PROCESS</td>
<td>No</td>
<td>Normal</td>
<td>MCS_LIST</td>
<td>Read/write</td>
</tr>
</tbody>
</table>
tag

---

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

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>MCS_prop_tag</td>
<td>MCS$r_prop_tag</td>
</tr>
<tr>
<td>Defined By</td>
<td>NAD$K_ATT_TAG</td>
<td></td>
</tr>
<tr>
<td>Required With <strong>new</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_LONGINT</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Write-once-at-creation</td>
<td></td>
</tr>
</tbody>
</table>

---

toolName—Tool Name

Names the tool that was running when an EVENT record was created.

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tag</td>
<td>MCS_prop_toolName</td>
<td>MCS$r_prop_toolName</td>
</tr>
<tr>
<td>Defined By</td>
<td>NAD$K_ATT_HISTORY_PRODUCT</td>
<td></td>
</tr>
<tr>
<td>Required With <strong>new</strong></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Normal</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_STRING</td>
<td></td>
</tr>
<tr>
<td>Access</td>
<td>Read/write</td>
<td></td>
</tr>
</tbody>
</table>
**toolVersion**—Tool Version

Indicates the version number of the processor that executed as a result of this method invocation.

Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>C binding</th>
<th>OpenVMS binding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCS_prop_toolVersion</td>
<td>MCS$r_prop_toolVersion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tag</th>
<th>NAD$K_ATT_TOOLVERSION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defined By</td>
<td>METHOD_INVOCATION</td>
</tr>
<tr>
<td>Required With</td>
<td>new</td>
</tr>
<tr>
<td>Type</td>
<td>Normal</td>
</tr>
<tr>
<td>Data Type</td>
<td>MCS_STRING</td>
</tr>
<tr>
<td>Access</td>
<td>Write-once-at-creation</td>
</tr>
</tbody>
</table>

**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.
summary

symbolic name
c binding
openvms binding
MCS_prop_top
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

user name—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
openvms binding
MCS_prop_userName
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

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
Type                Normal
Data Type           MCS_SMALLINT
Access              Read/write

Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>nad$k_warn</td>
<td>If validation fails, place a message on the error stack but continue the operation.</td>
</tr>
<tr>
<td>nad$k_fail</td>
<td>If validation fails, the operation fails.</td>
</tr>
</tbody>
</table>

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
OpenVMS binding
Tag
Defined By
Required With new
Type
Data type
Access

MCS_prop_validationApply
MCS$r_prop_validationApply
NAD$K_ATT_VAL_APPLY
VALIDATION
No
Normal
MCS_SMALLINT
Read/write

Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>nad$k_val_new</td>
<td>Apply when creating an object.</td>
</tr>
<tr>
<td>nad$k_val_setprop</td>
<td>Apply when modifying an object.</td>
</tr>
<tr>
<td>nad$k_val_free</td>
<td>Apply when deleting an object.</td>
</tr>
<tr>
<td>nad$k_val_reserve</td>
<td>Apply when reserving an object.</td>
</tr>
<tr>
<td>nad$k_val_replace</td>
<td>Apply when replacing an object.</td>
</tr>
<tr>
<td>nad$k_val_new_prot</td>
<td>Apply when creating a new type.</td>
</tr>
<tr>
<td>nad$k_val_setprop_prop</td>
<td>Apply when modifying a type.</td>
</tr>
</tbody>
</table>

validationQuery—Query Buffer for Validation

Allows a validation to be defined through a query buffer. (Supplied for compatibility with Oracle CDD/Repository V4.)
Summary

validationQuery

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
### Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>nad$k_val_start</td>
<td>Run validation before operation.</td>
</tr>
<tr>
<td>nad$k_val_end</td>
<td>Run validation after operation.</td>
</tr>
<tr>
<td>nad$k_val_ci</td>
<td>Run validation only if call is coming from the CI.</td>
</tr>
<tr>
<td>nad$k_val_mcs</td>
<td>Run validation only if call is coming from the MCS interface.</td>
</tr>
<tr>
<td>nad$k_val_ci_mcs</td>
<td>Run validation if call is either from the CI or the MCS interface.</td>
</tr>
</tbody>
</table>

### versionable—Versionable Type

Indicates whether an element type is a subtype of `VERSION`.

#### Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>MCS_prop_versionable</th>
</tr>
</thead>
<tbody>
<tr>
<td>C binding</td>
<td>MCS$r_prop_versionable</td>
</tr>
<tr>
<td>OpenVMS binding</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Defined By</th>
<th>ELEMENT_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required With <strong>new</strong></td>
<td>No</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Computed</td>
</tr>
<tr>
<td><strong>Data Type</strong></td>
<td>MCS_BOOLEAN</td>
</tr>
<tr>
<td><strong>Access</strong></td>
<td>Read-only</td>
</tr>
</tbody>
</table>

#### Constant Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>The element type is a subtype of <code>VERSION</code>.</td>
</tr>
<tr>
<td>FALSE</td>
<td>The element type is not a subtype of <code>VERSION</code>.</td>
</tr>
</tbody>
</table>
### versionNum—Version Number

Contains the version number of a `VERSION` element.

#### Summary

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>Type</th>
<th>Defined By</th>
<th>Required With</th>
<th>Data Type</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>C binding</td>
<td></td>
<td>Version</td>
<td>new</td>
<td>MCS_LONGINT</td>
<td>Read-only</td>
</tr>
<tr>
<td>OpenVMS binding</td>
<td></td>
<td></td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCS_prop_versionNum</td>
<td></td>
<td></td>
<td></td>
<td>MCS_PROP_VERSIONNUM</td>
<td></td>
</tr>
<tr>
<td>MCS$r_prop_versionNum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Property Descriptions 3–93
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; those 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>
<thead>
<tr>
<th>Relation Type</th>
<th>To Owner</th>
<th>To Member</th>
<th>To All Owners</th>
<th>To All Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPOSITE_PART</td>
<td>hasParents&lt;sup&gt;1&lt;/sup&gt;</td>
<td>hasChildren&lt;sup&gt;1&lt;/sup&gt;</td>
<td>-none-</td>
<td>allChildren&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>DEPENDS_ON</td>
<td>dependencies&lt;sup&gt;1&lt;/sup&gt;</td>
<td>dependents&lt;sup&gt;1&lt;/sup&gt;</td>
<td>allDependencies&lt;sup&gt;1&lt;/sup&gt;</td>
<td>allDependents&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td>HAS_COMPUTED_PROPERTY</td>
<td>typesHaving-CompProp</td>
<td>compPropDef&lt;sup&gt;1&lt;/sup&gt;</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_CONTEXT</td>
<td>ppForContext</td>
<td>currContext&lt;sup&gt;1&lt;/sup&gt;</td>
<td>-none-</td>
<td>-none-</td>
</tr>
</tbody>
</table>

<sup>1</sup>A description of this property appears in Chapter 3.
<table>
<thead>
<tr>
<th>Relation Type</th>
<th>Owner</th>
<th>Member</th>
<th>All Owners</th>
<th>All Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAS_CURR_COLLECTION</td>
<td>ppForCollection</td>
<td>currCollection</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_DATATYPE</td>
<td>dataTypeUsers</td>
<td>dataType</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_DEFAULT_METHOD</td>
<td>typesUsingMethod</td>
<td>methods</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_MESSAGE</td>
<td>implementingMethods</td>
<td>implementsMessage</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_METHOD_USED</td>
<td>miUsingMethod</td>
<td>methodUsed</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_MSGARG</td>
<td>messagesHavingMsgarg</td>
<td>argSpec</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_MSG_SENT</td>
<td>miForMessage</td>
<td>msgSent</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_MSG_TARGET</td>
<td>miForTarget</td>
<td>msgTarget</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_PARENT</td>
<td>childPartitions</td>
<td>basePartition</td>
<td>parentPartition</td>
<td>allChildPartitions</td>
</tr>
<tr>
<td>HAS_POSTAMBLE</td>
<td>methodUsingPostamble</td>
<td>postamble</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_PREAMBLE</td>
<td>methodUsingPreamble</td>
<td>preamble</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_PROPERTY</td>
<td>typesHavingProp</td>
<td>propDef</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HASRELATED_PARTITION</td>
<td>partitionsRelatedTo</td>
<td>related</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_RELATION</td>
<td>legalOwners</td>
<td>ownsRelation</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HASRELATION_PROPERTY</td>
<td>typesHavingRelProp</td>
<td>relPropDef</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HAS_SUPERTYPE</td>
<td>subTypes</td>
<td>superTypes</td>
<td>allSubTypes</td>
<td>allSuperTypes</td>
</tr>
<tr>
<td>HAS_TOP_COLLECTION</td>
<td>contextHavingAsTop</td>
<td>top</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>HISTORY_LIST</td>
<td>historyOwner</td>
<td>history</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>IMPLEMENTS_METHOD</td>
<td>compPropUsing</td>
<td>implementsMethod</td>
<td>-none-</td>
<td>-none-</td>
</tr>
</tbody>
</table>

1A description of this property appears in Chapter 3.
2This relation type is not documented in Chapter 1.

(continued on next page)
<table>
<thead>
<tr>
<th>Relation</th>
<th>Type</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>To Owner</td>
<td>To Member</td>
</tr>
<tr>
<td>IMPLEMENTS_RELATION</td>
<td>relPropUsing</td>
<td>implementsRelation</td>
</tr>
<tr>
<td>INVOKES_TOOL</td>
<td>invokedBy</td>
<td>invokes</td>
</tr>
<tr>
<td>METHOD_INPUT²</td>
<td>derives¹</td>
<td>derivedFrom¹</td>
</tr>
<tr>
<td>METHOD_OUTPUT²</td>
<td>derives¹</td>
<td>derivedFrom¹</td>
</tr>
<tr>
<td>METHOD_PARAMETER²</td>
<td>-none-</td>
<td>-none-</td>
</tr>
<tr>
<td>OBJECT_VALIDATION²</td>
<td>validationForType</td>
<td>associatedValidations¹</td>
</tr>
<tr>
<td>OPENED_BY</td>
<td>openedBy¹</td>
<td>openedFiles¹</td>
</tr>
<tr>
<td>RELATION³</td>
<td>logicalMember¹</td>
<td>legalMembers¹</td>
</tr>
<tr>
<td>RESERVED_BY</td>
<td>checkout¹</td>
<td>reservedBy</td>
</tr>
</tbody>
</table>

¹A description of this property appears in Chapter 3.
²This relation type is not documented in Chapter 1.
³The 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

<table>
<thead>
<tr>
<th>Property</th>
<th>Defined By</th>
<th>Traverses</th>
<th>To</th>
<th>Data Type/Access Type</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>allChildPartitions</td>
<td>PARTITION</td>
<td>HAS_PARENT</td>
<td>AO</td>
<td>S/RO</td>
<td>allParentPartitions</td>
</tr>
<tr>
<td>allChildren</td>
<td>COMPOSITE</td>
<td>COMPOSITE_PART</td>
<td>AM</td>
<td>S/RO</td>
<td>-none-</td>
</tr>
<tr>
<td>allDependencies</td>
<td>VERSION</td>
<td>DEPENDS_ON</td>
<td>AO</td>
<td>S/RO</td>
<td>allDependents</td>
</tr>
<tr>
<td>allDependents</td>
<td>VERSION</td>
<td>DEPENDS_ON</td>
<td>AM</td>
<td>S/RO</td>
<td>allDependents</td>
</tr>
<tr>
<td>allDerivedFrom</td>
<td>VERSION</td>
<td>METHOD_PARAMETER</td>
<td>AM</td>
<td>S/RO</td>
<td>allDerivedFrom</td>
</tr>
<tr>
<td>allDerives</td>
<td>VERSION</td>
<td>METHOD_PARAMETER</td>
<td>AO</td>
<td>S/RO</td>
<td>allDerivedFrom</td>
</tr>
<tr>
<td>allParentPartitions</td>
<td>PARTITION</td>
<td>HAS_PARENT</td>
<td>AM</td>
<td>S/RO</td>
<td>allChildPartitions</td>
</tr>
<tr>
<td>allSubTypes</td>
<td>TYPE</td>
<td>HAS_SUPERTYPE</td>
<td>AO</td>
<td>S/RO</td>
<td>allSuperTypes</td>
</tr>
<tr>
<td>allSuperTypes</td>
<td>TYPE</td>
<td>HAS_SUPERTYPE</td>
<td>AM</td>
<td>S/RO</td>
<td>allSubTypes</td>
</tr>
<tr>
<td>argSpec</td>
<td>MESSAGE</td>
<td>HAS_MSGARG</td>
<td>M</td>
<td>S/RW</td>
<td>messagesHavingMsgarg</td>
</tr>
<tr>
<td>associatedValidations</td>
<td>ELEMENT_TYPE</td>
<td>OBJECT_VALIDATION2</td>
<td>M</td>
<td>S/RW</td>
<td>validationForType</td>
</tr>
<tr>
<td>basePartition</td>
<td>CONTEXT</td>
<td>HAS_PARENT</td>
<td>M</td>
<td>E/WOC</td>
<td>-none-</td>
</tr>
<tr>
<td>checkout</td>
<td>CONTEXT</td>
<td>RESERVED_BY</td>
<td>O</td>
<td>S/RO</td>
<td>reservedBy</td>
</tr>
<tr>
<td>childPartitions</td>
<td>PARTITION</td>
<td>HAS_PARENT</td>
<td>O</td>
<td>S/RO</td>
<td>parentPartitions</td>
</tr>
<tr>
<td>compPropDef</td>
<td>ELEMENT_TYPE</td>
<td>HAS_COMPUTEDPROPERTY</td>
<td>M</td>
<td>S/RW</td>
<td>typesHavingCompProp</td>
</tr>
<tr>
<td>compPropUsing</td>
<td>METHOD</td>
<td>IMPLEMENTS_METHOD</td>
<td>O</td>
<td>E/RW</td>
<td>implementsMethod</td>
</tr>
<tr>
<td>contextHavingAsTop</td>
<td>VERSION</td>
<td>HAS_TOP_COLLECTION</td>
<td>O</td>
<td>E/RW</td>
<td>top</td>
</tr>
<tr>
<td>currCollection</td>
<td>PERSISTENT_PROCESS</td>
<td>HAS_CURR_COLLECTION</td>
<td>M</td>
<td>E/RO</td>
<td>ppForCollection</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Property</th>
<th>Defined By</th>
<th>Traverses</th>
<th>To</th>
<th>Data Type/Access Type</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>currContext¹</td>
<td>PERSISTENT_PROCESS</td>
<td>HAS_CONTEXT</td>
<td>M</td>
<td>E/RW</td>
<td>ppForContext</td>
</tr>
<tr>
<td>dataType¹</td>
<td>PROPERTY_TYPE MSGARG</td>
<td>HAS_DATATYPE</td>
<td>M</td>
<td>E/WOC</td>
<td>dataTypeUsers</td>
</tr>
<tr>
<td>dependencies¹</td>
<td>DATA_TYPE</td>
<td>HAS_DATATYPE</td>
<td>O</td>
<td>S/RO</td>
<td>dataType¹</td>
</tr>
<tr>
<td>dependents¹</td>
<td>VERSION</td>
<td>DEPENDS_ON</td>
<td>O</td>
<td>S/RW</td>
<td>dependents¹</td>
</tr>
<tr>
<td>derivedFrom¹</td>
<td>VERSION</td>
<td>DEPENDS_ON</td>
<td>M</td>
<td>S/RW</td>
<td>derives¹</td>
</tr>
<tr>
<td>derives¹</td>
<td>METHOD_INVOCATION</td>
<td>METHOD_OUTPUT²</td>
<td>O</td>
<td>S/WOC</td>
<td>derivedFrom¹</td>
</tr>
<tr>
<td>hasChildren¹</td>
<td>COMPOSITE</td>
<td>COMPOSITE_PART</td>
<td>M</td>
<td>S/RO</td>
<td>hasParents¹</td>
</tr>
<tr>
<td>hasParents¹</td>
<td>VERSION</td>
<td>COMPOSITE_PART</td>
<td>O</td>
<td>S/RO</td>
<td>hasChildren¹</td>
</tr>
<tr>
<td>history¹</td>
<td>NAMED_ELEMENT</td>
<td>HISTORY_LIST²</td>
<td>M</td>
<td>S/RW</td>
<td>historyOwner</td>
</tr>
<tr>
<td>historyOwner</td>
<td>EVENT</td>
<td>HISTORY_LIST²</td>
<td>O</td>
<td>E/RW</td>
<td>history¹</td>
</tr>
<tr>
<td>implementingMethods</td>
<td>MESSAGE</td>
<td>HAS_MESSAGE</td>
<td>O</td>
<td>S/RW</td>
<td>implementingMethods</td>
</tr>
<tr>
<td>implementsMessage¹</td>
<td>METHOD</td>
<td>HAS_MESSAGE</td>
<td>M</td>
<td>E/RW</td>
<td>compPropUsing</td>
</tr>
<tr>
<td>implementsMethod¹</td>
<td>HAS_COMPUTED_PROPERTY</td>
<td>IMPLEMENTS_METHOD</td>
<td>M</td>
<td>E/RW</td>
<td>relPropUsing</td>
</tr>
<tr>
<td>implementsRelation¹</td>
<td>HAS_RELATION_PROPERTY</td>
<td>IMPLEMENTS_RELATION</td>
<td>M</td>
<td>E/RW</td>
<td>invokes¹</td>
</tr>
<tr>
<td>invokedBy</td>
<td>BINARY_TOOL</td>
<td>INVOKES_TOOL</td>
<td>O</td>
<td>S/RW</td>
<td>invokes¹</td>
</tr>
<tr>
<td>invokes²</td>
<td>TOOL</td>
<td>INVOKES_TOOL</td>
<td>M</td>
<td>E/RW</td>
<td>invokedBy</td>
</tr>
</tbody>
</table>

¹A description of this property appears in Chapter 3.
²This 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>
<thead>
<tr>
<th>Property</th>
<th>Defined By</th>
<th>Traverses To</th>
<th>Data Type/Access Type</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>legalMembers¹</td>
<td>RELATION_TYPE</td>
<td>RELATION_MEMBER</td>
<td>M</td>
<td>S/RW</td>
</tr>
<tr>
<td>legalOwners¹</td>
<td>RELATION_TYPE</td>
<td>HAS_RELATION</td>
<td>O</td>
<td>S/RW</td>
</tr>
<tr>
<td>messagesHavingMsgarg</td>
<td>MSGARG</td>
<td>HAS_MSGARG</td>
<td>O</td>
<td>S/RW</td>
</tr>
<tr>
<td>methods¹</td>
<td>ELEMENT_TYPE</td>
<td>HAS_DEFAULT_METHOD</td>
<td>M</td>
<td>S/RW</td>
</tr>
<tr>
<td>methodUsed¹</td>
<td>ATIS_METHOD_INVOC</td>
<td>HAS_METHOD_USED⁴</td>
<td>M</td>
<td>E/WOC</td>
</tr>
<tr>
<td>methodUsingPostamble</td>
<td>METHOD</td>
<td>HAS_POSTAMBLE</td>
<td>O</td>
<td>S/RW</td>
</tr>
<tr>
<td>methodUsingPreamble</td>
<td>METHOD</td>
<td>HAS_PREAMBLE</td>
<td>O</td>
<td>S/RW</td>
</tr>
<tr>
<td>miForMessage</td>
<td>MESSAGE</td>
<td>HAS_MSG_SENT²</td>
<td>O</td>
<td>S/RW</td>
</tr>
<tr>
<td>miForTarget</td>
<td>VERSION</td>
<td>HAS_MSG_TARGET²</td>
<td>O</td>
<td>S/RO</td>
</tr>
<tr>
<td>msgSent¹</td>
<td>ATIS_METHOD_INVOC</td>
<td>HAS_MSG_SENT²</td>
<td>M</td>
<td>E/WOC</td>
</tr>
<tr>
<td>msgTarget¹</td>
<td>ATIS_METHOD_INVOC</td>
<td>HAS_MSG_TARGET²</td>
<td>M</td>
<td>E/WOC</td>
</tr>
<tr>
<td>openedBy¹</td>
<td>CONTEXT</td>
<td>OPENED_BY</td>
<td>O</td>
<td>S/RO</td>
</tr>
<tr>
<td>openedFiles¹</td>
<td>CONTEXT</td>
<td>OPENED_BY</td>
<td>M</td>
<td>S/RO</td>
</tr>
<tr>
<td>ownsRelation¹</td>
<td>ELEMENT_TYPE</td>
<td>HAS_RELATION</td>
<td>M</td>
<td>S/RW</td>
</tr>
<tr>
<td>parentPartition¹</td>
<td>PARTITION</td>
<td>HAS_PARENT</td>
<td>M</td>
<td>E/WOC</td>
</tr>
<tr>
<td>partitionsRelatedTo</td>
<td>PARTITION</td>
<td>HASRELATED_PARTITION</td>
<td>O</td>
<td>S/RO</td>
</tr>
<tr>
<td>postamble¹</td>
<td>METHOD</td>
<td>HAS_POSTAMBLE</td>
<td>M</td>
<td>S/RW</td>
</tr>
<tr>
<td>preamble¹</td>
<td>METHOD</td>
<td>HAS_PREAMBLE</td>
<td>M</td>
<td>S/RW</td>
</tr>
</tbody>
</table>

¹A description of this property appears in Chapter 3.
²This 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>
<thead>
<tr>
<th>Property</th>
<th>Defined By</th>
<th>Traverses</th>
<th>To</th>
<th>Data Type/ Access Type</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>ppForCollection</td>
<td>COLLECTION</td>
<td>HAS_CURR_ COLLECTION</td>
<td>O</td>
<td>E/RO</td>
<td>currCollection</td>
</tr>
<tr>
<td>ppForContext</td>
<td>CONTEXT</td>
<td>HAS_CONTEXT</td>
<td>O</td>
<td>E/RW</td>
<td>currContext</td>
</tr>
<tr>
<td>propDef¹</td>
<td>ELEMENT_TYPE</td>
<td>HAS_PROPERTY</td>
<td>M</td>
<td>S/RW</td>
<td>typesHavingProp</td>
</tr>
<tr>
<td>related¹</td>
<td>PARTITION</td>
<td>HASRELATED_PARTITION</td>
<td>M</td>
<td>S/RW</td>
<td>partitionsRelatedTo</td>
</tr>
<tr>
<td>relationMember¹</td>
<td>ELEMENT_TYPE</td>
<td>RELATION_MEMBER</td>
<td>O</td>
<td>S/RW</td>
<td>legalMembers</td>
</tr>
<tr>
<td>relPropDef¹</td>
<td>ELEMENT_TYPE</td>
<td>HASRELATIONPROPERTY</td>
<td>M</td>
<td>S/RW</td>
<td>typesHavingRelProp</td>
</tr>
<tr>
<td>relPropUsing</td>
<td>RELATION_TYPE</td>
<td>IMPLEMENTS_RELATION</td>
<td>O</td>
<td>E/RW</td>
<td>implementsRelation</td>
</tr>
<tr>
<td>reservedBy</td>
<td>VERSION</td>
<td>RESERVED_BY</td>
<td>M</td>
<td>E/RO</td>
<td>checkout</td>
</tr>
<tr>
<td>subTypes¹</td>
<td>TYPE</td>
<td>HAS_SUPER_TYPE</td>
<td>O</td>
<td>S/RO</td>
<td>superTypes</td>
</tr>
<tr>
<td>superTypes¹</td>
<td>ELEMENT_TYPE</td>
<td>HAS_SUPER_TYPE</td>
<td>M</td>
<td>S/WOC</td>
<td>subTypes</td>
</tr>
<tr>
<td>top¹</td>
<td>CONTEXT</td>
<td>HAS_TOP_COLLECTION</td>
<td>M</td>
<td>E/RW</td>
<td>contextHavingAsTop</td>
</tr>
<tr>
<td>typesHavingCompProp</td>
<td>PROPERTY_TYPE</td>
<td>HASCOMPUTEDPROPERTY</td>
<td>O</td>
<td>S/RW</td>
<td>compPropDef</td>
</tr>
<tr>
<td>typesHavingProp</td>
<td>PROPERTY_TYPE</td>
<td>HAS_PROPERTY</td>
<td>O</td>
<td>S/RW</td>
<td>propDef</td>
</tr>
<tr>
<td>typesHavingRelProp</td>
<td>PROPERTY_TYPE</td>
<td>HASRELATIONPROPERTY</td>
<td>O</td>
<td>S/RW</td>
<td>relPropDef</td>
</tr>
<tr>
<td>typesUsingMethod</td>
<td>METHOD</td>
<td>HAS_DEFAULT_METHOD</td>
<td>O</td>
<td>S/RW</td>
<td>methods</td>
</tr>
<tr>
<td>validationForType</td>
<td>VALIDATION</td>
<td>OBJECT_VALIDATION</td>
<td>O</td>
<td>E/RW</td>
<td>associatedValidations</td>
</tr>
</tbody>
</table>

¹A 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.
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**

- Element type
- Element type appears in more than one IIM Diagram
- Relation type
- Owner
- Member
- Arrow points to Supertype
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
A

ACAS_METHOD_INVOC element type
description, 1–5
access property
description, 3–3
accessType property
description, 3–5
AGGREGATE element type
description, 1–7
aliases property
description, 3–5
allCheckouts property
description, 3–6
allChildPartitions property
description, 3–7
allChildren property
description, 3–7
allDependencies property
description, 3–8
allDependents property
description, 3–9
allDerivedFrom property
description, 3–9
allDerives property
description, 3–10
allElementTypes property
description, 3–11
allInstances property
description, 3–11
allParentPartitions property
description, 3–12

allSubTypes property
description, 3–12
allSuperTypes property
description, 3–13
alternateNames property
description, 3–14
application property
description, 3–14
argList property
description, 3–15
argSpec property
description, 3–15
argsSent property
description, 3–15
associatedValidations property
description, 3–16
ATIS_METHOD_INVOC element type
description, 1–10
attach message
description, 2–5
method defined, 2–6
method refined
by COLLECTION, 2–6
attachment property
description, 3–17
attachmentInContext property
description, 3–17
autopurge property
description, 3–18
availlVersion property
description, 3–19
B

basePartition property
   description, 3–20
baseType property
   description, 3–21
baseTypeSize property
   description, 3–22
BINARY element type
   description, 1–12
BINARY_TOOL element type
   description, 1–15
branchName property
   description, 3–22
build message
   description, 2–7
   method defined, 2–8

C

checkout property
   description, 3–23
childPartitions property
   description, 3–23
close message
   description, 2–9
   method defined, 2–10, 2–11
   method refined
      by BINARY, 2–10
      by COLLECTION, 2–10
COLLECTION element type
   description, 1–18
Comments
   passing with messages, 2–1
COMPOSITE element type
   description, 1–21
COMPOSITE_PART element type
   description, 1–24
compPropDef property
   description, 3–24
CONTEXT element type
   description, 1–26

contextDir property
   description, 3–25
contextName property
   description, 3–25
control message
   description, 2–12
   method defined, 2–13
controlled property
   description, 3–26
conventions
   EER diagram, B–1
CPUTime property
   description, 3–26
createdDate property
   description, 3–27
currCollection property
   description, 3–28
currContext property
   description, 3–28

database element type
   description, 1–28
dataType property
   description, 3–29
DATA_TYPE element type
   description, 1–30
defaultAccess property
   description, 3–30
defaultAttachment property
   description, 3–31
definedLegalMembers property
   description, 3–31
definedLegalOwners property
   description, 3–32
definedMethods property
   description, 3–33
definedPropDefs property
   description, 3–33
deltaFile property
   description, 3–34
dependencies property
   description, 3–34
dependents property
description, 3–35
DEPENDS_ON element type
description, 1–33
derivedFrom property
description, 3–36
derives property
description, 3–36
description property
description, 3–37
detach message
description, 2–15
method defined, 2–16
method refined
  by BINARY, 2–16
  by COLLECTION, 2–16
differences message
description, 2–18
method defined, 2–19
direction property
description, 3–38
DIRECTORY element type
description, 1–35

E
edit message
description, 2–20
method defined, 2–21
method refined
  by TEXT, 2–21
elapsedTime property
description, 3–39
ELEMENT element type
description, 1–37
elementType property
description, 3–39
ELEMENT_TYPE element type
description, 1–38
EVENT element type
description, 1–41
export message
description, 2–22
method defined, 2–23
method refined
  method refined (cont’d)
    by BINARY, 2–23
    by COLLECTION, 2–23

F
filePath property
description, 3–40
firstVersion property
description, 3–41
flavor property
description, 3–17
free message
description, 2–24
disallowed
  by DATABASE, 2–26
  by ELEMENT_TYPE, 2–27
  by TYPE, 2–28
method defined, 2–27
method refined
  by BINARY, 2–26
  by COLLECTION, 2–26
  by CONTEXT, 2–26
  by DIRECTORY, 2–26
  by PARTITION, 2–27
  by PERSISTENT_PROCESS, 2–28
  by RELATION, 2–28
  by VERSION, 2–28
freeze message
description, 2–29
method defined, 2–30
freezeTime property
description, 3–41
funcType property
description, 3–42

G
getProp message
description, 2–31
method defined, 2–33
H

hasChildren property
description, 3–43

hasParents property
description, 3–44

hasComputedProperty element type
description, 1–43

hasContext element type
description, 1–45

hasCurrCollection element type
description, 1–47

hasDatatype element type
description, 1–49

hasDefaultMethod element type
description, 1–51

hasMessage element type
description, 1–53

hasMsgArg element type
description, 1–55

hasParent element type
description, 1–57

hasPostamble element type
description, 1–59

hasPreamble element type
description, 1–61

hasProperty element type
description, 1–63

hasRelatedPartition element type
description, 1–65

hasRelation element type
description, 1–67

hasRelationProperty element type
description, 1–69

hasSupertype element type
description, 1–71

hasTopCollection element type
description, 1–73

history property
description, 3–45

historyComment property
description, 3–45

I

IIM diagrams, B–1 to B–4

implementsMessage property
description, 3–46

implementsMethod property
description, 3–46

implementsRelation property
description, 3–47

implementsMethod element type
description, 1–75

implementsRelation element type
description, 1–77

import message
description, 2–34
method defined, 2–35
method refined
by binary, 2–35
by collection, 2–35

importedFrom property
description, 3–48

inPartition property
description, 3–48

instances property
description, 3–49

invocationStatus property
description, 3–49

invocationString property
description, 3–50

invokes property
description, 3–50

invokesTool element type
description, 1–79

K

keepHist property
description, 3–51
L

lastVersion property
  description, 3–52
legalMembers property
  description, 3–53, 3–73
legalOwners property
  description, 3–53, 3–65
logFile property
  description, 3–54

M

merge message
  description, 2–36
  method defined, 2–40
  method refined
    by composite, 2–39
    by text, 2–40
MESSAGE element type
  description, 1–81
messageName property
  description, 3–55
METHOD element type
  description, 1–84
methods property
  description, 3–55
methodType property
  description, 3–56
methodUsed property
  description, 3–56
METHOD_INVOCATION element type
  description, 1–87
MSGARG element type
  description, 1–89
msgSent property
  description, 3–57
msgTarget property
  description, 3–57
mutable property
  description, 3–58

N

name property
  description, 3–59
NAMED_ELEMENT element type
  description, 1–92
new message
  description, 2–42
  disallowed
    by database, 2–46
    by data_type, 2–46
  method defined, 2–46
  method refined
    by binary, 2–44
    by context, 2–45
    by directory, 2–46
    by element_type, 2–47
    by event, 2–47
    by message, 2–47
    by method, 2–48
    by named_element, 2–48
    by partition, 2–48
    by property_type, 2–49
    by relation, 2–49
    by relation_type, 2–50
    by version, 2–50
nextVersions property
  description, 3–60
node property
  description, 3–60
numChildren property
  description, 3–61

O

open message
  description, 2–52
  method defined, 2–53, 2–54
  method refined
    by binary, 2–53
    by collection, 2–53
openedBy property
  description, 3–62
openedFiles property
description, 3-62
OPENED_BY element type
description, 1-94
optionsString property
description, 3-63
OSVersion property
description, 3-64
owner property
description, 3-64
ownsRelation property
description, 3-65

P
parentInContext property
description, 3-65
parentPartition property
description, 3-66
PARTITION element type
description, 1-96
partitionDir property
description, 3-66
passingMechanism property
description, 3-67
path property
description, 3-68
pattern property
description, 3-68
PERSISTENT_PROCESS element type
description, 1-98
postamble property
description, 3-69
preamble property
description, 3-69
prevVersions property
description, 3-69
processingName property
description, 3-70
promote message
description, 2-55
method defined, 2-57
method refined
by BINARY, 2-56
by COLLECTION, 2-56

promote message (cont'd)
by MESSAGE, 2-57
by METHOD, 2-57
by TYPE, 2-57

propDef property
description, 3-71
PROPERTY_TYPE element type
description, 1-100

preamble property
description, 2-59
method defined, 2-60

R
referenceCount property
description, 3-72
related property
description, 3-73
relation element type
description, 1-103
relationMember property
description, 3-73
RELATION_MEMBER element type
description, 1-105
RELATION_TYPE element type
description, 1-107
relMember property
description, 3-74
relOwner property
description, 3-75
relPropDef property
description, 3-75
rename message
description, 2-61
disallowed
by DIRECTORY, 2-62
method defined, 2-62
replace message
description, 2-63
method defined, 2-66
method refined
by BINARY, 2-65
by COLLECTION, 2-65
by MESSAGE, 2-65
by METHOD, 2-66
replace message
    method refined (cont’d)
        by TYPE, 2–66
required property
description, 3–76
reserve message
description, 2–68
    method defined, 2–72
    method refined
        by BINARY, 2–70
        by COLLECTION, 2–70
        by MESSAGE, 2–71
        by METHOD, 2–71
        by TYPE, 2–71
reserved_by element type
description, 1–110
rootBranchInstances property
description, 3–77
rootBranchName property
description, 3–77
rootPath property
description, 3–78
rootVersion property
description, 3–79

S
scale property
description, 3–79
scalingFactor property
description, 3–80
setProp message
description, 2–74
    method defined, 2–77
    method refined
        by CONTEXT, 2–76
        by DEPENDS_ON, 2–76
        by PARTITION, 2–77
        by PERSISTENT_PROCESS, 2–78
        by VERSION, 2–78
simpleName property
description, 3–80
status property
description, 3–81

storedIn property
description, 3–82
storeType property
description, 3–82
subTypes property
description, 3–83
superTypes property
description, 3–84
symbols property
description, 3–85

T
tag property
description, 3–86
TEXT element type
description, 1–112
TEXT_TOOL element type
description, 1–115
TOOL element type
description, 1–118
toolName property
description, 3–86, 3–87
toolVersion property
description, 3–87
top property
description, 3–87
translate message
description, 2–79
    method defined, 2–80
    TYPE element type
description, 1–121

U
unfreeze message
description, 2–81
    method defined, 2–82
unreserve message
description, 2–83
    method defined, 2–85
    method disallowed
        by MESSAGE, 2–85
        by METHOD, 2–85
        by TYPE, 2–85
    method refined
unreserve message
   method refined (cont’d)
      by BINARY, 2-84
      by COLLECTION, 2-85
update message
   description, 2-87
   method defined, 2-87
      by COMPOSITE, 2-88
userName property
   description, 3-88

V

VALIDATION element type
   description, 1-124
validationAction property
   description, 3-89
validationApply property
   description, 3-89
validationQuery property
   description, 3-90
validationWhen property
   description, 3-91
verify message
   description, 2-89
   method defined, 2-92
   method refined
      by BINARY, 2-91
      by CONTEXT, 2-91
VERSION element type
   description, 1-127
versionable property
   description, 3-92
versionNum property
   description, 3-93