

Allegro[®] X Constraint Manager SKILL Guide

**Product Version 23.1
September 2023**

© 2023 Cadence Design Systems, Inc. All rights reserved.

Portions © Apache Software Foundation, Sun Microsystems, Free Software Foundation, Inc., Regents of the University of California, Massachusetts Institute of Technology, University of Florida. Used by permission. Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Allegro Platform Products contain technology licensed from, and copyrighted by: Apache Software Foundation, 1901 Munsey Drive Forest Hill, MD 21050, USA © 2000-2005, Apache Software Foundation. Sun Microsystems, 4150 Network Circle, Santa Clara, CA 95054 USA © 1994-2007, Sun Microsystems, Inc. Free Software Foundation, 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA © 1989, 1991, Free Software Foundation, Inc. Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, © 2001, Regents of the University of California. Daniel Stenberg, © 1996 - 2006, Daniel Stenberg. UMFPACK © 2005, Timothy A. Davis, University of Florida, (davis@cise.ulf.edu). Ken Martin, Will Schroeder, Bill Lorensen © 1993-2002, Ken Martin, Will Schroeder, Bill Lorensen. Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Massachusetts, USA © 2003, the Board of Trustees of Massachusetts Institute of Technology. All rights reserved.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information. Cadence is committed to using respectful language in our code and communications. We are also active in the removal and/or replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

1

<u>Introduction to Constraint Manager SKILL Functions</u>	5
<u>Writing a CM SKILL Program</u>	5
<u>Running a CM SKILL Program</u>	7
<u>Setting Up Constraint Manager SKILL</u>	9
<u>cmxIDBSkillInit</u>	9
<u>cmxIDBSkillExit</u>	10
<u>Single Object APIs</u>	11
<u>cmxIFindObject</u>	11
<u>cmxIFindOrCreateObject</u>	12
<u>cmxIFindOrCreateGroupObject</u>	13
<u>cmxIFindOrCreateCompositeObject</u>	14
<u>cmxIGetObjInfo</u>	15
<u>cmxIDeleteObject</u>	16
<u>cmxIRenameObject</u>	17
<u>cmxIAddObjectFlag</u>	18
<u>cmxIRemoveObjectFlag</u>	19
<u>cmxIAddAttributeFlag</u>	20
<u>cmxIRemoveAttributeFlag</u>	21
<u>cmxIHasAttributeFlag</u>	22
<u>Multiple Objects APIs</u>	23
<u>cmxICopyObject</u>	23
<u>cmxIMerge</u>	24
<u>Query Object APIs</u>	26
<u>cmxIGetObjects</u>	26
<u>cmxIGetObjectNames</u>	27
<u>cmxIGetParents</u>	28
<u>Object relationship APIs</u>	29
<u>cmxIReferenceObject</u>	29
<u>cmxIDeReferenceObject</u>	30
<u>Constraint APIs</u>	31

Allegro X Constraint Manager SKILL Guide

<u>cmxIPutAttribute</u>	31
<u>cmxIGetAttribute</u>	33
<u>cmxIGetPropertyNames</u>	35
<u>File APIs</u>	38
<u>cmxIImportFile</u>	38
<u>cmxIExportFile</u>	40
<u>axIImportDFATxtFileToDFM</u>	43
<u>Allegro Constraint Manager Server APIs</u>	44
<u>cmxIIsServerInitialized</u>	44
<u>axICMDBInit</u>	45
<u>axICMDBExit</u>	46
<u>Miscellaneous APIs</u>	47
<u>cmxIParseName</u>	47
<u>Description</u>	47
<u>Arguments</u>	47
<u>cmxICompile</u>	48

2

<u>Constants</u>	49
<u>Data Types</u>	49
<u>Object Types</u>	49
<u>Object Flags/Domains</u>	52
<u>Return Codes</u>	52
<u>CMXL Options</u>	52

Introduction to Constraint Manager SKILL Functions

Constraint Manager SKILL is a programmable interface that provides access to the Constraint Manager data, including object information, relationships, and properties. Programs written in SKILL can be augmented using the CMXL functions outlined in this document, similar to using AXL functions in the Allegro SKILL interface.

Constraint Manager SKILL programs can be written for use in both Allegro PCB Editor and Design Entry HDL. While the SKILL code can be the same, there are differences between the tools to note.

Writing a CM SKILL Program

Constraint Manager SKILL programs must be saved in text files to be loaded and run as required. The wrapper functions differentiates between Allegro PCB Editor and Design Entry HDL, which is easier to manage.

For example, to write a function named `mySkillFunc` that takes a file name as a parameter, the main algorithm could be written into a file named `myfunc.il`. This code can be run by both PCB Editor and Design Entry HDL.

myfunc.il

```
procedure(mySkillFunc_main(fileName)
  let((res)
    ; Your CM SKILL code.
    ; Set res to indicate success or failure.
    if(success then
      res = ACNS_OK
    else
      res = ACNS_FAIL
    )
  )
)
```

Allegro X Constraint Manager SKILL Guide

Introduction to Constraint Manager SKILL Functions

A wrapper for each of Allegro PCB Editor and Design Entry HDL is needed to run them in the respective applications.

myfunc_allegro.il

```
procedure(mySkillFunc(fileName)
  let((design res)
    axlCMDBInit()
    design = cmxlFindObject(ACNS_DESIGN)
    when(design
      cmxlDBSkillInit(design)
      ; Call your CM SKILL code. Use of errmsg is recommended.
      res = errmsg(mySkillFunc_main(fileName))
      unless(res
        printf("Error %L" errmsg.errmsg)
      )
      cmxlDBSkillExit()
    )
    axlCMDBExit()
    res
  )
)
```

myfunc_dehdl.il

```
procedure(mySkillFunc(fileName)
  let((design res)
    design = cmxlFindObject(ACNS_DESIGN)
    when(design
      cmxlDBSkillInit(design)
      ; Call your CM SKILL code. Use of errmsg is recommended.
      res = errmsg(mySkillFunc_main(fileName))
      unless(res
        printf("Error %L" errmsg.errmsg)
      )
      cmxlDBSkillExit()
    )
    res
  )
)
```

Important

Functions `axlCMDBInit` and `cmx1DBSkillInit` can be expensive (performance) when running Constraint Manager SKILL code. It is best to minimize the number of these calls. It means wrap all your Constraint Manager SKILL code in a single Init/Exit.

Important

You must not add or delete cross-section layers in a layout design while Constraint Manager SKILL is active. Always exit when your command is finished and initialize again the next time you run one of your commands.

Important

Design Entry HDL SKILL does not have access to Allegro SKILL functions. Do not use Allegro SKILL (`axl`) functions in any SKILL programs that will be run with Design Entry HDL.

Running a CM SKILL Program

Constraint Manager SKILL programs can be run from Allegro PCB Editor and Design Entry HDL using the built in SKILL commands of each applications. The previous example can be run by entering the following commands in the corresponding application's command window.

Allegro PCB Editor

```
skill load "myfunc.il"

skill load "myfunc_allegro.il"

skill mySkillFunc "test_file.txt"
```

Design Entry HDL

```
__callSkillFun load "myfunc.il"

__callSkillFun load "myfunc_dehdl.il"

__callSkillFun mySkillFunc "test_file.txt"
```



Important

The Design Entry HDL command console window does not echo back the results of the SKILL commands. Use SKILL functions in the program to open and write to a file on disk if information needs to be reported.

Example

A detailed example using Constraint Manager SKILL is available in the installation hierarchy at

```
<installation_hierarchy>/share/pcb/examples/skill/CMSK
```

This example is a text file that demonstrates how to use Constraint Manager SKILL to automatically generate the differential pairs for a design from a set of patterns. Instructions are also provided for extending the example to other group object types. You can start with the `cmsk_readme.txt` file.

Setting Up Constraint Manager SKILL

cmxIDBSkillInit

```
cmxIDBSkillInit(  
    g_cmdbScopePtr  
)  
==> *_result
```

Description

This function initializes the Constraint Manager database for SKILL access.

Arguments

<code>g_cmdbScopePtr</code>	Constraint Manager database pointer for the scope (design or system) being initialized.
-----------------------------	---

Value Returned

<code>nil</code>	if failure
<code>t/1</code>	if successful

Note: If a call has a side-effect of changing the Constraint Manager database outside of SKILL, the Constraint Manager database must be reinitialized. For example,

```
cmxIDBSkillExit()  
  
cmxImportFile(designID "newtech.tcfx")  
  
cmxIDBSkillInit()
```

See Also

[cmxIDBSkillExit](#)

cmxIDBSkillExit

```
cmxIDBSkillExit(  
    g_cmdbScopePtr  
)  
==> *_result
```

Description

This function cleans-up the Constraint Manager database after SKILL access.

Arguments

<code>g_cmdbScopePtr</code>	Constraint Manager database pointer for the scope (design or system) being cleaned-up.
-----------------------------	--

Value Returned

<code>nil</code>	if failure
<code>t/1</code>	if successful

See Also

[cmxIDBSkillInit](#)

Single Object APIs

cmxFindObject

```
cmxFindObject (
    x_objKind
    [t_objName]
    [g_cmdbScopePtr]
)
==> g_cmdbPtr
```

Description

This function queries the Constraint Manager database and returns an object if it exists.

Arguments

x_objKind	Kind of object being queried. ACNS_<kind> enum.
t_objName	Name of object to query. If name is not provided and <ul style="list-style-type: none">■ x_objKind is ACNS_DESIGN: Function returns active (read/write) design.■ x_objKind is ACNS_SYSTEM: Function returns the active system.■ x_objKind is anything else: ilcNil is returned.
g_cmdbScopePtr	Constraint Manager database pointer for the scope being queried. This option is not required when the scope object is either design or system.

Value Returned

g_cmdbPtr	Constraint Manager database pointer if object exists.
nil	Otherwise.

See Also

[cmxFindOrCreateObject](#)

cmx1FindOrCreateObject

```
cmx1FindOrCreateObject(  
    g_cmdbScopePtr  
    x_objKind  
    t_objName  
)  
==> g_cmdbPtr
```

Description

This function queries the Constraint Manager database and returns an object if it exists or creates it if it does not.

Arguments

<code>g_cmdbScopePtr</code>	Constraint Manager database pointer for the scope being processed.
<code>x_objKind</code>	Kind of object being created or found. ACNS_<kind> enum.
<code>t_objName</code>	Name of object to create or find.

Value Returned

<code>g_cmdbPtr</code>	Constraint Manager database pointer of existing or new object.
<code>nil</code>	If object cannot be successfully created.

Note: To distinguish objects created by SKILL program from other objects, pair this command with `cmx1PutAttribute` as shown in the following example :

```
objID = cmx1FindObject(scopeID ACNS_PHYS_CLASS "NEWCLASS")  
unless(objID  
    objID = cmx1FindOrCreateObject(scopeID ACNS_PHYS_CLASS "NEWCLASS")  
    when(objID  
        cmx1PutAttribute(objID "CDS_SKILL_DEFINED" ACNS_BOOLEAN '(t))  
    )  
)
```

See Also

[cmx1FindObject](#), [cmx1FindOrCreateGroupObject](#), [cmx1FindOrCreateCompositeObject](#)

cmxIFindOrCreateGroupObject

```
cmxIFindOrCreateGroupObject (
    g_cmdbParentPtr
    x_objKind
    t_objName
    l_cmdbMemberPtrs
)
==> g_cmdbPtr
```

Description

This function queries the Constraint Manager database and returns a group object if it exists or creates it if does not. A group object contains other Constraint Manager objects. For example, ACNS_DIFFPAIR, ACNS_MATCHGROUP, and ACNS_NET_GROUP. If the group is found but its members are different than specified, the current members will be replaced by the new list of members.

Arguments

<code>g_cmdbParentPtr</code>	Constraint Manager database pointer for the parent or scope of the object being processed.
<code>x_objKind</code>	Kind of object being created or found. ACNS_<kind> enum.
<code>t_objName</code>	Name of object to create or find.
<code>l_cmdbMemberPtrs</code>	List of member objects used to create or find the group object. Objects that are not legal members of the group will be ignored. Differential pair will not be created with invalid members. (Differential pairs must be created from a single XNet or two Nets/XNets).

Value Returned

<code>g_cmdbPtr</code>	Constraint Manager database pointer of existing or new object.
<code>nil</code>	If object cannot be successfully created.

See Also

[cmxIFindOrCreateObject](#), [cmxIFindObject](#)

cmxIFindOrCreateCompositeObject

```
cmxIFindOrCreateObject(  
    g_cmdbParentPtr  
    x_objKind  
    l_cmdbPtrs  
)  
==> g_cmdbPtr
```

Description

This function queries the Constraint Manager database and returns a composite object if it exists or creates it if it does not.

A composite object is comprised of other Constraint Manager database objects. For example, ACNS_PINPAIR, ACNS_CLASS_CLASS, ACNS_SUBCLASS_SUBCLASS, ACNS_REGION_CLASS (not yet supported), and ACNS_REGION_CLASS_CLASS (not yet supported).

Arguments

<code>g_cmdbParentPtr</code>	Constraint Manager database pointer for the parent or scope of the object being processed.
<code>x_objKind</code>	Kind of object being created or found. ACNS_<kind> enum.
<code>l_cmdbPtrs</code>	List of objects used to create or find the composite object.

Value Returned

<code>g_cmdbPtr</code>	Constraint Manager database pointer of existing or new object.
<code>nil</code>	If object cannot be successfully created.

See Also

[cmxIFindOrCreateObject](#), [cmxIFindObject](#)

cmxIGetObjInfo

```
cmxIGetObjInfo(  
    g_cmdbPtr  
)  
==> l_result
```

Description

This function returns the details for a Constraint Manager object.

Arguments

<code>g_cmdbPtr</code>	Constraint Manager database pointer for object being queried.
------------------------	---

Arguments

None.

Value Returned

<code>l_result</code>	List containing (ACNS_<kind> <physical name> <db name>) for the object.
<code>ilcNil</code>	If object does not exist.

See Also

[cmxIFindObject](#)

cmxlDeleteObject

```
cmxlDeleteObject(  
    g_cmdbPtr  
)  
==> x_result
```

Description

This function deletes an object from Constraint Manager database.

Arguments

<code>g_cmdbPtr</code>	Constraint Manager database pointer for object being queried.
------------------------	---

Arguments

None.

Value Returned

<code>x_result</code>	ACNS_OK, if object is successfully deleted.
-----------------------	---

See Also

[cmxlFindOrCreateObject](#)

cmxIRenameObject

```
cmxIRenameObject(  
    g_cmdbPtr  
    t_newName  
)  
==> x_result
```

Description

This function renames an object.

Arguments

<code>g_cmdbPtr</code>	Constraint Manager database pointer for object being queried.
<code>t_newName</code>	New name for the object.

Arguments

None.

Value Returned

<code>x_result</code>	ACNS_OK, if object is successfully renamed.
-----------------------	---

See Also

[cmxIFindOrCreateObject](#)

cmxlAddObjectFlag

```
cmxlAddObjectFlag(  
    g_cmdbPtr  
    x_flag  
)  
==> t if successful
```

Description

This function sets a flag on a Constraint Manager database object.

Arguments

g_cmdbPtr	Constraint Manager database pointer for object being updated.
x_flag	Set ACNS_OBJECT_FLAGS: <ul style="list-style-type: none">■ ACNS_OBJECT_SPACING_DOMAIN: To flag Net Class for Spacing domain.■ ACNS_OBJECT_SN_SPACING_DOMAIN: To flag Net Class for Same Net Spacing domain.■ ACNS_OBJECT_PHYSICAL_DOMAIN: To flag Net Class for Physical domain.■ ACNS_OBJECT_ELECTRICAL_DOMAIN: To flag Net Class for Electrical domain.■ ACNS_OBJECT_READONLY: To flag object as readonly.

Value Returned

ACNS_OK	If successful.
ACNS_FAIL	If not successful.

See Also

[cmxlRemoveObjectFlag](#)

cmxlRemoveObjectFlag

```
cmxlRemoveObjectFlag(  
    g_cmdbPtr  
    x_flag  
)  
==> t if successfull
```

Description

This function clears a flag on a Constraint Manager database object.

Arguments

<code>g_cmdbPtr</code>	Constraint Manager database pointer for object being updated.
<code>x_flag</code>	ACNS_OBJECT_FLAGS to remove.

Value Returned

<code>ACNS_OK</code>	If successful.
<code>ACNS_FAIL</code>	If not successful.

See Also

[cmxlAddObjectFlag](#)

cmxIAddAttributeFlag

```
cmxIAddAttributeFlag(  
    g_cmdbPtr  
    t_attrName  
    x_Flags  
    [g_cmdbParentPtr]  
)  
==> *_result
```

Description

This function queries an object for an attribute and sets flags on the attribute.

Arguments

g_cmdbPtr	Constraint Manager database pointer for object being queried.
t_attrName	Name of attribute to update.
x_flag	One or more flags that can be updated: <ul style="list-style-type: none">■ ACNS_ATTRIBUTE_LOCKED: To lock the attribute.■ ACNS_ATTRIBUTE_ACC_FLATTENED: To mark the attribute as compiler generated.
g_cmdbParentPtr	Constraint Manager database pointer for parent of the object when same attribute can exist for different object or parent combinations. For example, Match Groups. This parameter may be omitted when no parent object is required to specify the attribute.

Value Returned

ACNS_OK	If successful.
ACNS_FAIL	If flag could not be added.

See Also

[cmxIRemoveAttributeFlag](#), [cmxIHasAttributeFlag](#)

cmxlRemoveAttributeFlag

```
cmxlRemoveAttributeFlag(  
    g_cmdbPtr  
    t_attrName  
    x_Flags  
    [g_cmdbParentPtr]  
)  
==> *_result
```

Description

This function queries an object for an attribute and removes flags from the attribute.

Arguments

g_cmdbPtr	Constraint Manager database pointer for object being queried.
t_attrName	Name of attribute to update.
x_flag	One or more flags that can be updated: <ul style="list-style-type: none">■ ACNS_ATTRIBUTE_LOCKED: To unlock the attribute.■ ACNS_ATTRIBUTE_ACC_FLATTENED: To unmark the attribute as compiler generated.
g_cmdbParentPtr	Constraint Manager database pointer for parent of the object when same attribute can exist for different object or parent combinations. For example, Match Groups. This parameter may be omitted when no parent object is required to specify the attribute.

Value Returned

ACNS_OK	If successful.
ACNS_FAIL	If flag could not be removed.

See Also

[cmxlAddAttributeFlag](#), [cmxlHasAttributeFlag](#)

cmxHasAttributeFlag

```
cmxHasAttributeFlag(  
    g_cmdbPtr  
    t_attrName  
    x_Flags  
    [g_cmdbParentPtr]  
)  
==> *_result
```

Description

This function queries an object for an attribute to determine if the attribute's flags are set.

Arguments

g_cmdbPtr	Constraint Manager database pointer for object being queried.
t_attrName	Name of attribute to query.
x_flag	One or more flags that can be updated: <ul style="list-style-type: none">■ ACNS_ATTRIBUTE_LOCKED: To check if the attribute is flagged.■ ACNS_ATTRIBUTE_ACC_FLATTENED: To check if the attribute is compiler generated.
g_cmdbParentPtr	Constraint Manager database pointer for parent of the object when same attribute can exist for different object or parent combinations. For example, Match Groups. This parameter may be omitted when no parent object is required to specify the attribute.

Value Returned

ACNS_OK	If successful.
ACNS_FAIL	If all flags are not set or attribute is not found.

See Also

[cmxAddAttributeFlag](#), [cmxRemoveAttributeFlag](#)

Multiple Objects APIs

cmxlCopyObject

```
cmxlCopyObject (
    g_cmdbCopyFromPtr
    g_cmdbCopyToPtr
)
==> t
```

Description

This function copies all constraints from one object to another.

Arguments

<code>g_cmdbCopyFromPtr</code>	Constraint Manager database pointer for the copy-from object.
<code>g_cmdbCopyToPtr</code>	Constraint Manager database pointer for the copy-to object.

Value Returned

<code>nil</code>	if failure
<code>t/1</code>	if successful

See Also

[cmxlFindOrCreateObject](#)

cmxIMerge

```
cmxIMerge (  
    g_cmdbPtrDst  
    g_cmdbPtrSrc  
    g_cmdbPtrBase  
    [l_options]  
)  
==> x_result
```

Description

This function merges and updates the constraints of a source object to a destination object using an optional base object.

Arguments

<code>g_cmdbPtrDst</code>	Constraint Manager database pointer for destination object.
<code>g_cmdbPtrSrc</code>	Constraint Manager database pointer for source object.
<code>g_cmdbPtrBase</code>	Constraint Manager database pointer for base object. Used for Diff3 processing and can be zero for Diff2.

Allegro X Constraint Manager SKILL Guide

Introduction to Constraint Manager SKILL Functions

`l_options`

Optional processing options, a list of lists:

- `(CMXL_SET_CONTENT <x_mask>)`: Controls what to merge.
- `(CMXL_ADD_CONTENT <x_bit>)`: Ensures specific content is merged.
- `(CMXL_REMOVE_CONTENT <x_bit>)`: Ensures specific content is not merged.
- `(CMXL_IMPORT_REPORT_NAME <t_fileName>)`: File name for merge report.
- `(CMXL_IMPORT_SHOW_REPORT)`: Automatically shows report after merge.
- `(CMXL_IMPORT_MODE <x_mode>)`: Sets the merge mode to
 - 0 - for Diff3
 - 1 - for merge
 - 2 - for overwrite
 - 3 - for replace
- `(CMXL_IMPORT_REPORT_ONLY)`: Reports the results of the import without doing the import.
- `(CMXL_PROP_NAME "<name>")`: Merges a specific property or constraint name.
- `(CMXL_UPDATE_MODE <mode>)`: Updates mode for Diff3 conflicts where destination values are different from base values.
 - 0 : Preserves all overrides/conflicts in the destination.
 - 1 : Updates all overrides/conflicts in the destination.

Value Returned

`x_result`

ACNS_RC return codes

Query Object APIs

cmxGetObject

```
cmxGetObject (
    g_cmdbParentPtr
    x_assocKind
)
==> l_results
```

Description

This function returns all child objects for a parent.

Arguments

<code>g_cmdbParentPtr</code>	Constraint Manager database pointer for the parent. For example, Net Class.
<code>x_assocKind</code>	Kind of objects to return. <code>ACNS_<kind></code> enum. <code>ACNS_NULL_OBJ</code> will return all the children of a parent.

Value Returned

<code>l_result</code>	List of Constraint Manager database child object pointers.
-----------------------	--

See Also

[cmxGetObjectNames](#), [cmxGetParents](#)

cmxlGetObjectNames

```
cmxlGetObjectNames (
    g_cmdbParentPtr
    x_assocKind
    [g_cmdbAssocDesignPtr]
)
==> l_results
```

Description

This function returns all child object names for a parent.

Use this function when needing all objects and these objects would not exist in the Constraint Manager database. For example, nets of a design.

Arguments

<code>g_cmdbParentPtr</code>	Constraint Manager database pointer for the parent, such as design.
<code>x_assocKind</code>	Kind of children to return. ACNS_<kind> enum. For example, net.
<code>g_cmdbAssocDesignPtr</code>	Optional. Constraint Manager database pointer for the children's design. This option is required when querying for the children of a system-level object. For example, nets of a system-level XNet.

Value Returned

<code>l_result</code>	List of child object names.
-----------------------	-----------------------------

See Also

[cmxlGetObjects](#), [cmxlGetParents](#)

cmxlGetParents

```
cmxlGetParents (
    g_cmdbPtr
    x_parentKind
    [x_traverseHierarchy]
    [x_filterMask]
)
==> l_results
```

Description

This function returns all parents for a given object.

Arguments

<code>g_cmdbPtr</code>	Constraint Manager database pointer for the object. For example, design.
<code>x_parentKind</code>	Kind of parent to return. ACNS_<kind> enum. For example, Net Group.
<code>x_traverseHierarchy</code>	1: Traverses hierarchy to find parent. By default, traverse is not performed.
<code>x_filterMask</code>	This option is only required when asking for a specific type of Net Classes. For example, electrical versus physical and spacing, and so on.

Value Returned

<code>l_result</code>	List of Constraint Manager database parent pointers.
-----------------------	--

See Also

[cmxlGetObjects](#)

Object relationship APIs

cmxlReferenceObject

```
cmxlReferenceObject(  
    g_cmdbParentPtr  
    g_cmdbChildPtr  
)  
==> x_result
```

Description

This function creates a relationship between a parent and child object. It adds an association between two objects.

Arguments

<code>g_cmdbParentPtr</code>	Constraint Manager database pointer for the parent. For example, Net Class or Constraint Set.
<code>g_cmdbChildPtr</code>	Constraint Manager database pointer for the child. For example, Net.

Value Returned

<code>x_result</code>	<ul style="list-style-type: none">■ <code>ACNS_OK</code>, if relationship is created or already exists.■ <code>ACNS_NULL</code>, if Constraint Set relationship already exists.■ <code>ACNS_FAIL</code>, if relationship could not be created.
-----------------------	--

See Also

[cmxlDeReferenceObject](#)

cmxIDeReferenceObject

```
cmxIDeReferenceObject(  
    g_cmdbParentPtr  
    g_cmdbChildPtr  
)  
==> x_result
```

Description

This function deletes a relationship between a parent and child object. It removes an association between two objects.

Arguments

<code>g_cmdbParentPtr</code>	Constraint Manager database pointer for the parent. For example, Net Class or Constraint Set.
<code>g_cmdbChildPtr</code>	Constraint Manager database pointer for the child. For example, net.

Value Returned

<code>x_result</code>	■ <code>ACNS_OK</code> , if relationship is removed or does not exist.
	■ <code>ACNS_NULL</code> , if Constraint Set relationship does not exist.
	■ <code>ACNS_FAIL</code> , if invalid objects are provided to the function.

See Also

[cmxIReferenceObject](#)

Constraint APIs

cmx1PutAttribute

This function sets an attribute on a Constraint Manager database object.

```
cmx1PutAttribute (
    g_cmdbPtr
    t_attrName
    x_dataType
    l_attrValue
    [g_cmdbParentPtr]
)
==> x_result
```

g_cmdbPtr	Constraint Manager database pointer for object being updated
t_attrName	Name of attribute to set.
x_dataType	Data type of l_attrValue: <ul style="list-style-type: none">■ ACNS_STRING/ACNS_ENUM: l_attrValue is a string.■ ACNS_INTEGER: l_attrValue is an integer.■ ACNS_DOUBLE: l_attrValue is a double integer.■ ACNS_BOOLEAN: l_attrValue is a boolean value (nil is false, any other value is true).■ ACNS_DOUBLE_ARRAY: l_attrValue is a list of double integers.■ ACNS_STRING_ARRAY/ACNS_ENUM_ARRAY : l_attrValue is a list of strings.■ ACNS_NULL_TYPE: l_attrValue is ignored and attribute is deleted from the object, if it exists.
l_attrValue	Attribute value to set.

Allegro X Constraint Manager SKILL Guide

Introduction to Constraint Manager SKILL Functions

`g_cmdbParentPtr`

Constraint Manager database pointer for the parent of an object when the same attribute can exist for different object/parent combinations. For example, Match Groups.

This parameter may be omitted when no parent object is required to specify the attribute.

Value Returned

`x_result`

ACNS_OK, if attribute is successfully set.

Examples

You can use this function to lock a net so that its members cannot be edit. To lock a net, set the `MEMBERSHIP_LOCKED` attribute to `ACC_DEFINED`. The following command locks the membership of Net Group NG1:

```
dsn = cmxlFindObject(ACNS_DESIGN);Returns active design.
```

```
ng1 = cmxlFindObject(ACNS_NET_GROUP, "NG1", dsn);Returns Net Group "NG1" in the active design.
```

```
cmxlPutAttribute(NG1 "MEMBERSHIP_LOCKED" ACNS_STRING ACC_DEFINED)
```

Related Topics

[cmxlGetAttribute](#)

cmxlGetAttribute

```
cmxlGetAttribute(  
    g_cmdbPtr  
    t_attrName  
    [x_dataType]  
    [x_traverseFlag]  
    [g_cmdbParentPtr]  
)  
==> *_result
```

Description

This function queries an (parent) object for an attribute and returns the value as requested.

Arguments

<code>g_cmdbPtr</code>	Constraint Manager database pointer for object being queried.
<code>t_attrName</code>	Name of attribute to query.
<code>x_dataType</code>	Alternate data type to return. Default is the data type of the attribute, but can be overridden to return: <ul style="list-style-type: none">■ <code>ACNS_STRING</code>: Returns value as a string.■ <code>ACNS_INTEGER</code>: Returns value as an integer.■ <code>ACNS_DOUBLE</code>: Returns value as a double integer.■ <code>ACNS_BOOLEAN</code>: Returns value as a boolean value (nil or 1).■ <code>ACNS_DOUBLE_ARRAY</code>: Returns value as list of double integers.■ <code>ACNS_STRING_ARRAY</code>: Returns value as list of strings.
<code>x_traverseFlag</code>	1 : traverse object hierarchy to find attribute. This is the default. 0 : Only look on object.
<code>g_cmdbParentPtr</code>	Constraint Manager database pointer for the parent of an object when the same attribute can exist for different object/parent combinations. For example, Match Groups. This parameter may be omitted when no parent object is required to specify the attribute.

Value Returned

`*_result` Attribute value as defined by `x_dataType`.

See Also

[cmxIPutAttribute](#)

cmxlGetPropertyNames

```
cmxlGetPropertyNames (
    x_objKind
    t_attrKind
    t_attrCategory
    t_domain)
==> l_attrNames
```

Description

This function determines all available properties that meet the specified criteria and returns a list of their names.

Arguments

<code>x_objKind</code>	Find only properties that are applicable to the specified object kind. Any ACNS_<object> type.
<code>t_attrKind</code>	String specifying the attribute type to look for: <ul style="list-style-type: none">■ "property": Find property types■ "constraint": Find all constraints■ "reds_null_attr_kind": Do not filter by attribute type. May also use the predefined ACNS_NULL_ATTR_KIND symbol.

Allegro X Constraint Manager SKILL Guide

Introduction to Constraint Manager SKILL Functions

`t_attrCategory`

String specifying the attribute category to look for:

- `clocks`: Find only Clock related properties
- `crosstalk`: Find only Crosstalk related properties
- `shielding`: Find only Shielding related properties
- `ringing`: Find only Ringing related properties
- `delay`: Find only Propagation Delay related properties
- `length`: Find only Length related properties
- `emi`: Find only Emissions related properties
- `routing`: Find only Routing related properties
- `spacing`: Find only Spacing related properties
- `match`: Find only Match Group related properties
- `powerrail`: Find only Power Rail related properties
- `user`: Find only User Defined properties
- `reds_null_category`: Do not filter by attribute category.

May also use the predefined `ACNS_NULL_CATEGORY` symbol.

Allegro X Constraint Manager SKILL Guide

Introduction to Constraint Manager SKILL Functions

`t_domain`

Find only properties defined for the specified domain

- `electrical`: Find only Electrical attributes
- `physical`: Find only Physical attributes
- `spacing`: Find only Spacing attributes
- `sn_spacing`: Find only Same Net Spacing attributes
- `assembly`: Find only Assembly attributes
- `powerintegrity`: Find only Power Integrity attributes
- `property`: Find only Property domain attributes
- `ilc`: Find only ILC attributes
- `dff`: Find only DFF attributes
- `dfa`: Find only DFA attributes
- `nodomain`: Find only attributes without a domain specified.
- `anydomain`: Do not filter by attribute domain.

May also use the predefined `ACNS_ANY_ATTR_DOMAIN` symbol.

Value Returned

`l_attrNames`

List of strings containing the names of each property found.

File APIs

cmxIImportFile

```
cmxIImportFile(  
    g_cmdbPtr  
    t_fileName  
    [l_options]  
)  
==> x_result
```

Description

This function imports a file to update a Constraint Manager object.

Arguments

<code>g_cmdbPtr</code>	Constraint Manager database pointer for object being updated.
<code>t_fileName</code>	Name of file to import/read.
<code>l_options</code>	Processing options (list of lists): <ul style="list-style-type: none">■ <code>(CMXL_SET_CONTENT <x_mask>)</code>: Controls what to import.■ <code>(CMXL_ADD_CONTENT <x_bit>)</code>: Ensures specific content is imported.■ <code>(CMXL_REMOVE_CONTENT <x_bit>)</code>: Ensures specific content is not imported.■ <code>(CMXL_IMPORT_REPORT_NAME <t_fileName>)</code>: File name for import report.■ <code>(CMXL_IMPORT_SHOW_REPORT)</code>: Automatically shows report after import.■ <code>(CMXL_IMPORT_MODE <x_mode>)</code>: Sets the import mode to:<ul style="list-style-type: none">□ 0 - for Diff3□ 1 - for Merge□ 2 - for Overwrite□ 3 - for Replace■ <code>(CMXL_IMPORT_INIT_CONTENT_FROM_FILE)</code>: Sets the content mask from the input file.■ <code>(CMXL_IMPORT_REPORT_ONLY)</code>: Reports the results of the import without doing the import.

Value Returned

<code>x_result</code>	ACNS_RC return codes.
-----------------------	-----------------------

See Also

[cmxlExportFile](#)

cmxlExportFile

```
cmxlExportFile(  
    g_cmdbPtr  
    t_fileName  
    [l_options]  
)  
==> x_result
```

Description

This function exports a file for a Constraint Manager object.

Arguments

g_cmdbPtr	Constraint Manager database pointer for object being exported.
t_fileName	Name of the file to export.
l_options	Processing options: <ul style="list-style-type: none">■ (CMXL_SET_CONTENT <x_mask>): Controls what to export.■ (CMXL_ADD_CONTENT <x_bit>): Ensures specific content is exported.■ (CMXL_REMOVE_CONTENT <x_bit>): Ensures specific content is not exported.

Notes

The CMXL_<variable>_CONTENT parameters allows you to override the default content in the technology file.

- CMXL_SET_CONTENT: Ignores the Contents entry in an input technology file and uses specified MASK. A MASK is collection of BITS. The BIT value needs to be converted from binary to decimal before passing it to the function.
- CMXL_ADD_CONTENT: Ignores the Contents entry from an input technology file and uses specified BIT. The BIT value should be provided in decimal format.
- CMXL_REMOVE_CONTENT: Can only be used if CMXL_IMPORT_INIT_CONTENT_FROM_FILE option is already specified. Use this

Allegro X Constraint Manager SKILL Guide

Introduction to Constraint Manager SKILL Functions

option to disable processing of specific Content entries. The BIT value should be provided in decimal format.

The BITs supported are as follows:

Bits Supported

Bit	Description
0x00000001	Process the cross-section.
0x00000002	Process the electrical constraint information.
0x00000004	Process the spacing constraint information.
0x00000008	Process the physical constraint information.
0x00000010	Process the user defined property definitions.
0x00000100	Process the analysis mode information.
0x00000200	Process the worksheet customization information.
0x00000400	Process the functions (formulas/measurements/predicates).
0x00001000	Process the Same Net Spacing constraint information.
0x00002000	Process the Assembly Constraint information.
0x00008000	Process the Properties domain information.
	Note: Properties from other domains will not be processed.
0x00010000	Process only cross-section neutral data (Generic layers).
0x00020000	Process the power integrity property information
0x00040000	Process the Manufacturing ILC constraints.
0x00100000	Process the Manufacturing DFF constraints.
0x00200000	Process the Manufacturing DFA constraints.
0x00400000	Process the Manufacturing DFT constraints.

Value Returned

`x_result` ACNS_RC return codes.

See Also

cmxlImportFile

Examples

- Only process electrical constraint data using SET, regardless of the contents of the input file.

```
Skill> elecContent = list(CMXL_SET_CONTENT 2)
Skill> dsn = cmxlFindObject(ACC_DESIGN)
Skill> cmxlImportFile( dsn, "<fileName>", list(elecContent))
```

- Process spacing and physical constraint data using SET, regardless of the contents of the input file.

```
Skill> contentMask = 3 + 4
Skill> physSpcContent = list(CMXL_SET_CONTENT contentMask)
Skill> dsn = cmxlFindObject(ACC_DESIGN)
Skill> cmxlImportFile( dsn, "<fileName>", list(physSpcContent))
```

- Process spacing and physical constraint data using ADD, regardless of the contents of the input file.

```
Skill> physContent = list(CMXL_ADD_CONTENT 4)
Skill> spcContent = list(CMXL_ADD_CONTENT 3)
Skill> dsn = cmxlFindObject(ACC_DESIGN)
Skill> cmxlImportFile( dsn, "<fileName>", list(physContent, spcContent))
```

- Do not process sssembly constraint data using REMOVE.

```
Skill> notAssemContent = list(CMXL_REMOVE_CONTENT 8192)
Skill> dsn = cmxlFindObject(ACC_DESIGN)
Skill> cmxlImportFile( dsn, "<fileName>",
list(list(CMXL_IMPORT_INIT_CONTENT_FROM_FILE), notAssemContent))
```

axlImportDFATxtFileToDFM

```
axlImportDFATxtFileToDFM(`  
    t_fileName  
    `)  
==> x_result
```

Description

This function imports the DFA package to package constraints file into the DFM package to package rules.

The CSets, created for the DFM package to package spacing checks are based on the information available in the input .dfa file. Any existing DFM to DFA package to package constraints are removed from the design. After importing the file, the generated CSets are required to be re-assigned against the specific layers in the corresponding design worksheet.

Note: It is recommended to import the DFM package to package rules using File - Import - Technology File. Use this function only when creating a design with an existing .dfa file.

Arguments

<code>t_fileName</code>	Name of the .dfa file to be imported.
-------------------------	---------------------------------------

Value Returned

<code>x_result</code>	Returns <code>nil</code> if failure and <code>t</code> if successful
-----------------------	--

Examples

```
axlImportDFATxtFileToDFM("c:/testcases/const.dfa")
```

Allegro Constraint Manager Server APIs

cmxIsServerInitialized

```
cmxIsServerInitialized()  
==> *_result
```

Description

This function determines if Constraint Manager database is initialized.

Arguments

None

Value Returned

*_result	1; if initialize
nil	Otherwise

axlCMDBInit

```
axlCMDBInit()  
==> x_result
```

Description

This function initializes Constraint Manager database server. It is available only in Allegro PCB Editor.

Arguments

None

Value Returned

<code>x_result</code>	0; if successful
-----------------------	------------------

axlCMDBExit

```
axlCMDBInit()  
==> x_result
```

Description

This function shuts down the Constraint Manager database server. It is available only in Allegro PCB Editor.

Arguments

None

Value Returned

<code>x_result</code>	0; if successful
-----------------------	------------------

Miscellaneous APIs

cmxlParseName

```
cmxlParseName (
    t_name
    t_suffix
    [t_prefix]
)
==> l_result
```

Description

This function parses a name based upon a prefix/suffix to find the base name and bit number.

Arguments

t_name	Name of the file to export.
t_suffix	Suffix which must exist in the name.
t_prefix	Prefix which must exist in the name.

Value Returned

l_result	List containing (name and bit number). <ul style="list-style-type: none">■ name: base name (string) without prefix, suffix, and bit number.■ bit: bit number (integer) if it exists. -1 if no bit number was found.
----------	--

cmxlCompile

```
cmxlCompile(  
    g_cmdbPtr  
    t_configFileName  
    [t_reportName]  
)  
==> x_result
```

Description

This function runs the ACC compiler based upon a configuration file.

Arguments

<code>g_cmdbPtr</code>	Constraint Manager database pointer for the object being updated.
<code>t_configFileName</code>	Name of XML configuration file. See <code>acc_input.xsd</code> .
<code>t_reportName</code>	Optional name of XML report (must have <code>.xml</code> extension).

Value Returned

<code>ACNS_OK</code>	if successfull.
<code>ACNS_FAIL</code>	if unsuccessful.

Constants

Data Types

The following constants represent the various data types used in the Constraint Manager database.

ACNS_DOUBLE

ACNS_STRING

ACNS_BOOLEAN

ACNS_ENUM

ACNS_INTEGER

ACNS_DOUBLE_ARRAY

ACNS_STRING_ARRAY

ACNS_ENUM_ARRAY

ACNS_INTEGER_ARRAY

Object Types

The following constants represent the various object type supported in the Constraint Manager database.

ACNS_SYSTEM

ACNS_DESIGN

ACNS_PARTDEFN

ACNS_PARTINST

ACNS_GATEDEFN

ACNS_GATEINST

Allegro X Constraint Manager SKILL Guide

Constants

ACNS_PINDEFN
ACNS_PININST
ACNS_XNET
ACNS_NET
ACNS_LAYER
ACNS_REGION
ACNS_ECSET
ACNS_GROUP
ACNS_DRC
ACNS_CLASS
ACNS_DIFFPAIR
ACNS_BUS
ACNS_PINPAIR
ACNS_MATCHGROUP
ACNS_RESULT_PINPAIR
ACNS_RESULT
ACNS_ECSET_MATCHGROUP
ACNS_ECSET_PINPAIR
ACNS_ECSET_PININST
ACNS_CLINESEG
ACNS_DESIGNINST
ACNS_ELECTRICAL_DRC_GROUP
ACNS_SPACING_DRC_GROUP
ACNS_PHYSICAL_DRC_GROUP
ACNS_DESIGN_DRC_GROUP
ACNS_EXTERNAL_DRC_GROUP
ACNS_PCSET
ACNS_SCSET
ACNS_CLASS_CLASS

Allegro X Constraint Manager SKILL Guide

Constants

ACNS_REGION_CLASS
ACNS_REGION_CLASS_CLASS
ACNS_LAYERSET
ACNS_RATBUNDLE
ACNS_SNSET
ACNS_SN_SPACING_DRC_GROUP
ACNS_NET_GROUP
ACNS_WIRE_PROF
ACNS_ACSET
ACNS_PARTINST_CLASS
ACNS_ASSEMBLY_DRC_GROUP
ACNS_PICSET
ACNS_POWERRAIL
ACNS_GENERIC_LAYER
ACNS_SUBCLASS
ACNS_SUBCLASS_SUBCLASS

The following constants represent the various sub object types that can be used instead of ACNS_CLASS.

ACNS_ELEC_CLASS
ACNS_PHYS_CLASS
ACNS_SPC_CLASS
ACNS_SNSPC_CLASS
ACNS_PHYS_REGION_CLASS
ACNS_SPC_REGION_CLASS
ACNS_SNSPC_REGIONCLASS

Object Flags/Domains

The following constants represent the various flags, including domains that are supported in Constraint Manager database.

ACNS_OBJECT_POWERINTEGRITY_DOMAIN

ACNS_OBJECT_ASSEMBLY_DOMAIN

ACNS_OBJECT_ELECTRICAL_DOMAIN

ACNS_OBJECT_SPACING_DOMAIN

ACNS_OBJECT_SN_SPACING_DOMAIN

ACNS_OBJECT_PHYSICAL_DOMAIN

ACNS_OBJECT_READONLY

Return Codes

The following codes may be returned from various methods.

Flags	Return Code	Status
ACNS_OK	1	Successful
ACNS_NULL	0	Operation not performed
ACNS_FAIL	-1	Severe failure
ACNS_NOT_SUPPORTED	-22	Operation not supported
ACNS_EXISTS	-7	Object already exists
ACNS_NOT_FOUND	-4	Query not found

CMXL Options

The following options are used to control the information processed by file or object merge functions.

CMXL_SET_CONTENT	Controls what to process.
CMXL_ADD_CONTENT	Ensures specific content is process.
CMXL_REMOVE_CONTENT	Ensures specific content is not processed.

Allegro X Constraint Manager SKILL Guide

Constants

CMXL_IMPORT_REPORT_NAME	File name for import/merge report
CMXL_IMPORT_SHOW_REPORT	Automatically shows report after import/merge
CMXL_IMPORT_MODE	Sets the import mode to Merge, Overwrite, or Replace
CMXL_IMPORT_INIT_CONTENT_FROM_FILE	Sets the content mask from the input file
CMXL_IMPORT_REPORT_ONLY	Reports the results of the import/merge without doing the import
CMXL_UPDATE_MODE	Sets the update mode for handling Diff3 conflicts during merge
CMXL_PROP_NAME	Sets a specific property/constraint to merge